A semi-analytic galaxy formation code.

# Contents

## I. Installation and Basic Use

1. About Galacticus  
   1.0.1. License .................................................. 4

2. Installation  
   2.1. Binaries ....................................................... 5  
   2.2. Installation Scripts ......................................... 5  
   2.3. Required Libraries and Tools  
      2.3.1. Essential Requirements ................................ 5  
      2.3.2. Typical Requirements .................................. 8  
      2.3.3. Full Requirements ...................................... 10  
   2.4. Compiling Galacticus  
      2.4.1. Compiling with OpenMP Parallelism .................... 13  
   2.5. Installing Without Root Access  
   2.6. Installing on Mac OS X  
      2.6.1. Update GNU compilers .................................. 17  
      2.6.2. Installing HDF5, FoX and FGSL ......................... 18  
      2.6.3. Installing Perl Modules ................................ 19

3. Running Galacticus  
   3.1. Configuration File ........................................... 21  
   3.2. Parameter Files  
      3.2.1. Validating Parameter Files  
      3.2.2. Generating Parameter Files ................................ 22  
   3.3. Running Galacticus  
      3.3.1. Writing Data To a Temporary File ...................... 23  
      3.3.2. Restarting A Crashed Run  
      3.3.3. Running Grids of Models ................................ 24  
      3.3.4. Analysis of Models  
      3.3.5. Running Models in “Embarrassingly Parallel” Mode .................... 30  
      3.3.6. Limiting the Load Average ................................ 31  
   3.4. Additional Codes  
      3.4.1. Excursion_Sets ....................................... 31  
      3.4.2. Halo_Mass_Functions .................................... 32  
      3.4.3. Power_Spectra ......................................... 34

4. Input Parameters ................. 37
5. Extracting and Analyzing Results

5.1. General Structure of Output File

5.1.1. UUID
5.1.2. Build Information
5.1.3. Parameters
5.1.4. Version
5.1.5. globalHistory
5.1.6. Outputs
nodeData group
mergerTree datasets
mergerTree subgroups
5.1.7. Optional Outputs
Redshifts
Mass Accretion Histories
Merger Tree Dump
Conditional Mass Functions
Pre-Evolution Merger Trees
5.2. Perl Module for Data Extraction
5.2.1. Derived Properties
Available Derived Properties
5.2.2. Galaxy Clustering via the Halo Model
5.3. Topics in Analysis of GALACTICUS Outputs
5.3.1. Building Volume Limited Samples
Building Redshift Catalogs
5.4. Postprocessing Scripts
5.5. Reprocessing Through Dust Using Grasil
5.5.1. Using the Galacticus::Grasil Module
5.6. Meta-Analysis of GALACTICUS
5.6.1. Tree Construction/Evolution Timing
5.6.2. ODE Evolver Profiler
5.7. Meta-Data in Plots
5.8. Perl Statistics Modules
5.8.1. Statistics::Histograms

6. Input Data

6.1. Broadband Filters
6.2. Merger Trees
6.2.1. Processing of Merger Tree Files
Requirements for GALACTICUS Input Parameters
6.2.2. Setting of Halo Properties
Dark Matter Scale Radii
Satellite Merger Times
Dark Matter Halo Spins

7. Tutorials

7.1. Running GALACTICUS on N-body Merger Trees
7.1.1. Setting Input Parameters
Node Positions
Virial Orbits
Merging Times and Targets
### 7.1.2. Analyzing the Output

- Positions and Velocities
- Subhalo Masses

### 7.2. Generating Mock Catalogs with Lightcones from the Millennium Simulation

### 7.3. Using the Instantaneous Recycling Approximation

### 7.4. Computing Dust Attenuation and Emission Using Galacticus+Grasil

### 7.5. Outputting Stellar Luminosities

#### 7.5.1. Postprocessing of Stellar Spectra

#### 7.5.2. Migrating Parameter Files to a New Version

#### 7.5.3. Computing Emission Lines

#### 7.5.4. Reionization Calculations

### 8. Troubleshooting

- Low CPU utilization with large numbers of output redshifts

### 9. Numerical Implementation

#### 9.1. Timestepping Criteria

- Tree Criteria
- "Branch Segment" Criteria
- "Parent" Criteria
- "Satellite" Criteria
- "Sibling" Criteria
- "Mergee" Criteria

#### 9.1.2. General Criteria

- "Simple" Criteria
- "Satellite" Criteria

#### 9.1.3. Output Criteria

- "History" Criteria
- "Main Branch Evolution" Criteria

### 10. Constraining Galacticus

#### 10.1. Model Accuracy

#### 10.2. Model Convergence

#### 10.3. Model Discrepancy

##### 10.3.1. Monte Carlo Merger Trees

##### 10.3.2. Fixed Virial Orbits

##### 10.3.3. Jiang et al. (2008) Merger Time Scatter

#### 10.4. Optimal Halo Mass Function Sampling

##### 10.4.1. Li & White (2009) Stellar Mass Function

##### 10.4.2. Refining by Other Merger Tree Statistics
III. Physical Implementation

11. Definitions and Conventions Used in Galacticus
11.1. Luminosity Units
11.2. Peculiar Velocities

12. Node Components
12.1. (Supermassive) Black Hole
  12.1.1. “Standard” Implementation
    Properties
    Initialization
    Differential Evolution
    Event Evolution
    Additional Output
  12.1.2. “Simple” Implementation
    Properties
    Initialization
    Differential Evolution
    Event Evolution
    Additional Output

12.2. Dynamics Statistics
  12.2.1. “Bars” Implementation
    Properties
    Initialization
    Differential Evolution
    Event Evolution
    Additional Output

12.3. Hot Halo
  12.3.1. “Very Simple” Implementation
    Properties
**Indices**

- **Properties**
- **Initialization**
- **Differential Evolution**
- **Event Evolution**

**Inter-Output Quantities**

- **Properties**
- **Initialization**
- **Differential Evolution**
- **Event Evolution**

## Physical Implementations

13. **Accretion of Gas into Halos**
   - **Simple Method**
   - **Null Method**

13.2. **Cosmology Functions**
   - **Matter + Lambda**
   - **Matter + Dark Energy**

13.3. **Circumnuclear Accretion Disks**
   - **Shakura-Sunyaev Geometrically Thin, Radiatively Efficient Disks**
   - **Advection Dominated, Geometrically Thick, Radiatively Inefficient Flows (ADAFs)**
   - **“Switched” Disks**
   - **Eddington-limited Disks**

13.4. **Dark Matter Structure Formation**
   - **Primordial Power Spectrum**
     - (Running) Power Law Spectrum
   - **Cosmological Mass Root Variance**
     - Filtered Power Spectrum
   - **Power Spectrum Variance Window Function**
     - Top-hat
     - Sharp in $k$-space
     - Hybrid of Top-hat and Sharp in $k$-space
   - **Non-linear Matter Power Spectrum**
     - Linear
     - Peacock & Dodds (1996)
     - CosmicEmu
   - **Transfer Function**
     - Null
     - BBKS
     - Eisenstein & Hu
     - CAMB
     - File
   - **Linear Growth Function**
     - Simple
   - **Critical Overdensity**
     - Spherical Collapse (Matter + Cosmological Constant)
13.4.8. Critical Overdensity Mass Scaling .................................................. 247
    Null ................................................................. 247
    Warm Dark Matter ............................................... 247
13.4.9. Virial Density Contrast ................................................................. 248
    Fixed ................................................................. 248
    Spherical Collapse (Matter + Cosmological Constant) .......... 248
    Kitayama & Suto (1996) ........................................... 248
    Friends-of-Friends .............................................. 248
13.4.10. Halo Bias ......................................................................................... 249
    Press-Schechter ..................................................... 249
    Sheth-Tormen ......................................................... 249
    Tinker .................................................................. 249
13.4.11. Halo Mass Function ........................................................................... 249
    Press-Schechter ..................................................... 249
    Sheth-Tormen ......................................................... 249
    Tinker .................................................................. 249
13.5. Conditional Mass Functions ..................................................................... 250
13.5.1. Behroozi et al. (2010) ....................................................................... 250
13.6. Cooling of Gas Inside Halos .................................................................... 250
13.6.1. Cooling Function .................................................................................. 250
    Atomic Collisional Ionization Equilibrium Using CLOUDY .......... 250
    Collisional Ionization Equilibrium From File ....................... 251
    CMB Compton Cooling .................................................... 251
    Molecular Hydrogen (Galli-Palla) ......................................... 251
13.6.2. Cooling Rate ......................................................................................... 252
    Cole et al. (2000) ....................................................................... 252
    Simple ........................................................................ 252
    Simple Scaling ................................................................ 252
    White & Frenk ............................................................. 253
13.6.3. Cooling Rate Modifier ......................................................................... 253
    Cut-Off ....................................................................... 253
13.6.4. Cooling Radius ..................................................................................... 253
    Simple ........................................................................ 253
    Isothermal .................................................................. 253
13.6.5. Cooling: Freefall Radius ....................................................................... 254
    Dark Matter Halo .......................................................... 254
13.6.6. Cooling: Infall Radius .......................................................................... 254
    Cooling Radius ........................................................... 254
    Cooling and Freefall Radii ............................................... 254
    Dark Matter Halo .......................................................... 254
13.6.7. Cooling Specific Angular Momentum .................................................. 254
    Constant Rotation ........................................................ 254
    Mean .......................................................................... 255
13.6.8. Cooling Time ......................................................................................... 255
    Simple ........................................................................ 255
13.6.9. Time Available for Cooling ................................................................... 255
    Halo Formation ................................................................ 255
White & Frenk (1991) .......................................................... 255

13.6.10. Time Available for Freefall During Cooling
Halo Formation .............................................................. 256

13.7. Cosmology .............................................................. 256

13.7.1. Matter + Cosmological Constant Universes ................. 256

13.8. Dark Matter Halos ..................................................... 256

13.8.1. Mass Accretion History ........................................... 256
Wechsler et al. (2002) .................................................... 256
Zhao et al. (2009) .......................................................... 256

13.8.2. Density Profile ..................................................... 257
Isothermal ................................................................. 257
NFW ................................................................. 257
Einasto ................................................................. 257

13.8.3. Dark Matter Halo Profile Concentration .................... 257
Navarro, Frenk & White (1996) ......................................... 258
Gao (2008) ................................................................. 258
Zhao (2009) ................................................................. 258
Muñoz-Cuartas (2011) .................................................... 259
Prada et al. (2011) ......................................................... 259
Dutton & Macciò (2014) .................................................. 259
Diemer & Kravtsov (2014) .............................................. 260
WDM ................................................................. 260

13.8.4. Density Profile Shape ............................................. 260
Gao (2008) ................................................................. 260

13.8.5. Mass Loss Rates .................................................. 261
Null ................................................................. 261
van den Bosch et al. (2005) ............................................. 261

Power Law ............................................................... 261
Halo Mass Function ...................................................... 262
Stellar Mass Function .................................................... 262

13.8.7. Spin Parameter Distribution .................................. 262
Lognormal ............................................................... 262
Bett et al. (2007) ........................................................ 262
Delta Function ............................................................ 262

13.9. Disk Stability / Bar Formation ................................. 262

13.9.1. Null ................................................................. 263

13.9.2. Efstathiou, Lake & Negroponte ............................. 263

13.9.3. Efstathiou, Lake & Negroponte + Tidal Forces ............. 263

13.10. Excursion Set Barrier ............................................. 263

13.10.1. Linear Barrier ................................................... 264

13.10.2. Quadratic Barrier .............................................. 264

13.10.3. Critical Overdensity Barrier ............................... 264

13.11. Excursion Set Barrier First Crossing Distribution ........ 264

13.11.1. Linear Barrier ................................................... 264

13.11.2. Farahi ............................................................ 264

13.11.3. Zhang & Hui (2006) .......................................... 266

13.12 Excursion Set Barrier Remapping
   - Null Remapping
   - Scale Remapping
   - Sheth-Mo-Tormen Remapping

13.13 Galactic Structure
   - Fixed
   - Linear
   - Simple
   - Adiabatic

13.14 Galactic Structure Initial Radii
   - Static
   - Adiabatic

13.15 Galaxy Merging
   - Mass Movements
     - Very Simple
     - Simple
     - Baugh et al. (2005)
   - Remnant Sizes
     - Null
     - Cole et al. (2000)
     - Covington et al. (2008)
   - Progenitor Properties
     - Cole et al. (2000)
     - Standard

13.16 Hot Halo Mass Distribution
   - $\beta$-profile
   - Ricotti (2000)
   - Null

13.17 Hot Halo Mass Distribution Profile: Cored Profile Core Radius
   - Growing Core
   - Virial Fraction

13.18 Hot Halo Ram Pressure Force
   - Null
   - Font et al. (2008)

13.19 Hot Halo Ram Pressure Stripping Radius
   - Virial Radius
   - Font et al. (2008)

13.20 Hot Halo Ram Pressure Stripping Timescale
   - Halo Dynamical Timescale
   - Ram Pressure Acceleration

13.21 Hot Halo Temperature Profile
   - Virial Temperature

13.22 Initial Mass Functions
   - Initial Mass Function Selection
     - Fixed
     - Disk and Spheroid
   - Baugh et al. (2005) Top-Heavy
   - Chabrier
13.35 Satellite Tidal Fields ................................................................. 292
  13.35.1 Null ........................................................................ 292
  13.35.2 Spherical Symmetry .......................................................... 292
13.36 Satellite Tidal Stripping ............................................................. 292
  13.36.1 Null ........................................................................ 292
  13.36.2 Zentner (2005) .............................................................. 293
13.37 Satellite Tidal Heating ............................................................... 293
  13.37.1 Null ........................................................................ 293
  13.37.2 Gnedin (1999) .............................................................. 293
13.38 Star Formation Rate Surface Densities ........................................... 293
  13.38.1 Kennicutt-Schmidt ......................................................... 294
  13.38.2 Extended Schmidt .......................................................... 294
  13.38.3 Blitz-Rosolowsky ........................................................... 294
  13.38.4 Krumholz-McKee-Tumlinson ............................................ 295
13.39 Star Formation Timescales .......................................................... 295
  13.39.1 Fixed ....................................................................... 295
  13.39.2 Halo Scaling ................................................................. 296
  13.39.3 Dynamical Time ............................................................. 296
  13.39.4 Integrated Surface Density .............................................. 296
  13.39.5 Baugh et al. (2005) ....................................................... 296
13.40 Stellar Population Properties ....................................................... 296
  13.40.1 Instantaneous ................................................................. 297
  13.40.2 Noninstantaneous ........................................................... 297
13.41 Stellar Population Spectra ........................................................... 297
  13.41.1 Conroy, White & Gunn (2009) ........................................... 297
  13.41.2 File ........................................................................ 298
13.42 Stellar Population Spectra Postprocessing ....................................... 298
  13.42.1 Inoue, Shimizu & Iwata (2014) IGM Attenuation ................. 298
  13.42.2 Meiksin (2006) IGM Attenuation ...................................... 298
  13.42.3 Madau (1995) IGM Attenuation ....................................... 298
  13.42.4 Lyman-continuum Suppression ......................................... 298
  13.42.5 Recent Star Formation .................................................. 298
  13.42.6 Identity .................................................................. 298
13.43 Stellar Astrophysics ................................................................. 298
  13.43.1 Basics .......................................................... 299
    File ........................................................................ 299
  13.43.2 Stellar Winds ............................................................... 299
    Leitherer et al. (1992) ......................................................... 299
  13.43.3 Stellar Tracks ............................................................... 300
    File ........................................................................ 300
  13.43.4 Supernovae Type Ia ........................................................ 300
    Nagashima et al. (2005) Prescription ...................................... 300
  13.43.5 Population III Supernovae ................................................ 300
    Heger & Woosley (2002) ...................................................... 301
  13.43.6 Stellar Feedback ........................................................... 301
    Standard .................................................................. 301
13.44 Substructure and Merging ............................................................ 301
  13.44.1 Merging Timescales ........................................................ 301
    Lacey & Cole (1993) ......................................................... 301
IV. Development

15. Developing GALACTICUS

15.1. Getting Started .................................................. 325
  15.1.1. Using BitBucket ............................................. 325
15.2. Making Simple Changes ......................................... 325
  15.2.1. Contributing Your Changes Back To GALACTICUS
          Via E-mail ................................................................ 326
          Using BitBucket .................................................... 326
15.3. Making Bigger Changes ............................................. 326
15.4. Releases ............................................................... 327
  15.4.1. Bug Fixes In Releases ......................................... 327

16. Coding GALACTICUS ......................................................... 329

16.1. Numerical Tools ..................................................... 329
  16.1.1. Finding Roots of Equations ................................. 329
16.2. Computation Dependencies and Data Files ................. 331
16.3. Optimization .......................................................... 332
  16.3.1. Unique IDs and Stored Properties ....................... 332
16.4. Mixed Language Coding ......................................... 332
  16.4.1. Component Property Methods ............................ 333
  16.4.2. Using Functions in C++ .................................... 333
  16.4.3. Adding New Implementations in C++ ................. 333
16.5. Objects ................................................................. 334
  16.5.1. Enumerations .................................................. 334
          tfamily adjustElements ........................................ 334
          tfamily componentType ........................................ 334
          tfamily coordinateSystem ..................................... 334
          tfamily dataType .................................................. 334
          tfamily extrapolationType ..................................... 335
          tfamily massDistributionSymmetry ......................... 335
          tfamily massType ............................................... 335
          tfamily metaDataType .......................................... 335
          tfamily metallicityScale ....................................... 335
          tfamily propertyType ........................................... 336
          tfamily rangeExpand ............................................ 336
          tfamily rangeExpandSignExpect ............................. 337
tfamily inputParameterList ........................................ 365
tfamily integerScalarHash ....................................... 365
tfamily intergalacticMediumStateClass ......................... 366
tfamily intergalacticMediumStateFile ......................... 366
tfamily intergalacticMediumStateInternal ..................... 367
tfamily intergalacticMediumStateRecFast ..................... 368
tfamily intergalacticMediumStateSimple ....................... 368
tfamily keplerOrbit ............................................. 369
tfamily longIntegerHistory ...................................... 371
tfamily massDistribution ......................................... 371
tfamily massDistributionBetaProfile .......................... 372
tfamily massDistributionCylindrical ......................... 372
tfamily massDistributionHernquist ............................ 373
tfamily massDistributionNFW .................................. 373
tfamily massDistributionSersic .................................. 374
tfamily massDistributionSpherical ............................ 374
tfamily mergerTree ............................................... 375
tfamily mergerTreeData ......................................... 375
tfamily mergerTreeImporterClass .............................. 376
tfamily mergerTreeImporterGalacticus ......................... 377
tfamily mergerTreeImporterSussing ............................ 379
tfamily nodeComponent ........................................... 380
tfamily nodeComponentAgeStatistics ......................... 381
tfamily nodeComponentAgeStatisticsNull ...................... 385
tfamily nodeComponentAgeStatisticsStandard ............... 388
tfamily nodeComponentBasic .................................... 392
tfamily nodeComponentBasicNonEvolving ...................... 395
tfamily nodeComponentBasicNull ................................ 399
tfamily nodeComponentBasicStandard ......................... 402
tfamily nodeComponentBasicStandardTracking .............. 406
tfamily nodeComponentBlackHole ................................ 409
tfamily nodeComponentBlackHoleNull ......................... 413
tfamily nodeComponentBlackHoleSimple ....................... 418
tfamily nodeComponentBlackHoleStandard .................... 422
tfamily nodeComponentDarkMatterProfile .................... 426
tfamily nodeComponentDarkMatterProfileNull ............... 429
tfamily nodeComponentDarkMatterProfileScale .............. 433
tfamily nodeComponentDarkMatterProfileScalePreset ........ 436
tfamily nodeComponentDarkMatterProfileScaleShape ........ 439
tfamily nodeComponentDisk ...................................... 443
tfamily nodeComponentDiskExponential ...................... 449
tfamily nodeComponentDiskNull .................................. 456
tfamily nodeComponentDiskVerySimple ....................... 463
tfamily nodeComponentDynamicsStatistics .................... 469
tfamily nodeComponentDynamicsStatisticsBars .............. 472
tfamily nodeComponentDynamicsStatisticsNull ............. 474
tfamily nodeComponentFormationTime ......................... 477
tfamily nodeComponentFormationTimeCole2000 .............. 479
tfamily nodeComponentFormationTimeNull .................... 481
17. Adding New Methods

17.1. Code Directives

17.2. Identifying Components and Mass Types

17.3. Components

17.3.1. Component Structure

17.3.2. Extending Components

17.3.3. Implementing a New Component

Component Definition

Component Initialization

Component Access, Creation and Destruction

Component Methods

Component Evolution

Evolution Interrupts

17.4. Existing Method Types

17.4.1. Functions

Accretion Disk Spectra
21. Acknowledgements

VI. Appendices

A. Merger Tree File Format
   A.1. Basic File Format ........................................ 2000
   A.1.1. Flexibility and Extensibility .......................... 2000
   A.1.2. A Note on Scalar Attributes ........................... 2000
   A.1.3. Example File Structure ................................. 2000
   A.2. Cosmology Group ........................................... 2001
   A.2.1. Standard Attributes .................................... 2002
   A.3. Group Finder Group ....................................... 2002
   A.3.1. Standard Attributes .................................... 2003
   A.4. Simulation Group .......................................... 2003
   A.4.1. Standard Attributes .................................... 2005
   GADGET-specific Standard Attributes ........................ 2005
   A.5. Units Group .................................................. 2006
   A.6.1. Standard Attributes .................................... 2010
   A.6.2. Standard Datasets ....................................... 2011
   A.7. Tree Index Group ............................................ 2012
   A.7.1. Standard Datasets ....................................... 2013
   A.8. Merger Trees Group ......................................... 2013
   A.9. Particles Group ............................................. 2013
   A.10. Merger Tree Builder ...................................... 2015
   A.10.1. File Formats .......................................... 2018
   A.10.2. Exporting Trees from GALACTICUS .................... 2019

B. Plotting Support .................................................. 2021
   B.1. Plotting with Gnuplot ..................................... 2021
   B.2. Merger Tree Diagrams with Dot ........................... 2023

Glossary ............................................................... 2041

Acronyms ............................................................. 2043
Part I.

Installation and Basic Use
1. About Galacticus

**Galacticus** is a semi-analytic model of galaxy formation. It solves equations describing how galaxies evolve in a merging hierarchy of dark matter halos in a cold dark matter universe. **Galacticus** has much in common with other semi-analytic models, such as the range of physical processes included and the type of quantities that it can predict.

In designing **Galacticus** our main goal was to make the code flexible, modular and easily extensible. Much greater priority was placed on making the code easy to use and modify than on making it fast. We believe that a modular and extensible nature is crucial as galaxy formation is an evolving science. In particular, key design features are:

**Extensible methods for all functions:** Essentially all functions within **Galacticus** are designed to be extensible, meaning that you can write your own version and insert it into **Galacticus** easily. For example, suppose you want to use an improved functional form for the cold dark matter (CDM) halo mass function. You would simply write a subroutine conforming to a specified template that computes this mass function and add a short directive (see §17.1) in your code which explains to the build system how to insert this function in **Galacticus**. A recompile of the code will then incorporate your new function.

**Extensible components for tree nodes:** The basic structure in **Galacticus** is a merger tree, which consists of a linked tree of nodes which have various properties. **Galacticus** works by evolving the nodes forwards in time subject to a collection of differential equations and other rules. Each node can contain an arbitrary number of *components*. A component may be a dark matter halo, a galactic disk, a black hole etc. Each component may have an arbitrary number of *properties* (some of which may be evolving, others of which can be fixed). **Galacticus** makes it easy to add additional components. For example, suppose you wanted to add a “stellar halo” component (consisting of stars stripped from satellite galaxies). To do this, you would write a module which specifies the following for this component:

- Number of properties;
- Interfaces to set and get property values and rates of change;
- “Pipes” which allow for flows of mass/energy/etc. from one component to another;
- Routines describing the differential equations which govern the evolution of the properties;
- Routines describing how the component responds to various events (e.g. the node becoming a satellite, a galaxy-galaxy merger, etc.);
- Auxiliary routines for handling outputs etc.

Short directives embedded in this module explain to the **Galacticus** build system how to incorporate the new component. A recompile will then build your new component into **Galacticus**. Typically, a new component can be created quickly by copying an existing one and modifying it as necessary. Furthermore, multiple implementations of a component are allowed. For example, **Galacticus** contains a component which is a Hernquist spheroid. You could add a de Vaucouler’s spheroid component. A simple input parameter then allows you to select which implementation will be used in a given run.
1. About Galacticus

Centralized ODE solver: GALACTICUS evolves nodes in merger trees by calling an ODE solver which integrates forwards in time to solve for the evolution of the properties of each component in a node. This means that you do not need to provide explicit solutions for ODEs (in many cases such solutions are not available anyway) and timestepping is automatically handled to achieve a specified level of precision. The ODE solver allows for the evolution to be interrupted. A component may trigger an interrupt at any time and may do so for a number of reasons. A typical use is to actually create a component within a given node—for example when gas first begins to cool and inflow in a node the disk component must be created. Other uses include interrupting evolution when a merging event occurs.

1.0.1. License

Copyright 2009, 2010, 2011, 2012, 2013, Andrew Benson <abenson@obs.carnegiescience.edu>

GALACTICUS is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

GALACTICUS is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with GALACTICUS. If not, see <http://www.gnu.org/licenses/>.
2. Installation

2.1. Binaries

If you just want to run GALACTICUS, and don’t want to modify the code in any way you can try using a standalone binary copy of GALACTICUS. This is often a good way to try out GALACTICUS without the overhead of installing libraries and tools required to compile GALACTICUS. Binaries can be found at the download page and are currently available only for x86_64 Linux. The binary is updated every night to reflect any updates to GALACTICUS.

2.2. Installation Scripts

Installing all the tools required for GALACTICUS can take some time. To make this process easier we’ve developed some simple installation scripts for Ubuntu, Fedora and Red Hat flavors of Linux. You can find these at the download page, with discussion of their use at the installation scripts page. These should be run as root and will attempt to install everything need to run GALACTICUS. We suggest that you look through the scripts to see what they’re doing before running them—we make no guarantees that they won’t install something that breaks other aspects of your system—use them at your own risk!

2.3. Required Libraries and Tools

We run GALACTICUS on Fedora Linux primarily. Installation instructions below are general, but we give some examples that are specific to Fedora (they probably work on other yum-based Linux distributions though). For other distributions/OSs you’ll need to figure out how to install pre-built packages or else install from source. We list tools and libraries in three categories: “essential”, “typical” and “full”. Essential tools are those required to compile and run GALACTICUS. Typical tools includes those additionally needed for common analysis tasks. The full set of tools allows you to run any of the scripts/programs included with GALACTICUS. The order in which tools are listed is the suggested order for installation—some tools depend on others which should therefore be installed first.

2.3.1. Essential Requirements

Perl The Perl language is a part of every Linux distribution that we know of, so use whatever method your distribution prefers to install it if you don’t already have it. On Fedora the following (as root) will install Perl:

```
yum install perl
```

If you need to install from source visit http://www.perl.org/ and follow the instructions given there. We currently use Perl v5.12.4, although earlier versions may work (for compiling GALACTICUS we have had success with versions as early as 5.8.8—you can use perl -v to discover which version of Perl you have installed).
2. Installation

**GNU Make** Galacticus uses GNU Make to build executables. Other implementations of make may work, but we make no guarantees. On Fedora the following (as root) will install GNU Make:

```
yum install make
```

Alternatively, GNU Make can be downloaded and installed from source as described online.

**GFortran** Galacticus is written primarily in Fortran using various aspects of the Fortran 2003 specification. We use GNU Fortran to compile Galacticus. You can use other Fortran compilers, but we make no claims as to whether they will support all required features of the Fortran language, or that they will produce correctly working code. On Fedora the following (as root) will install GNU Fortran:

```
yum install gcc-gfortran
```

Alternatively, GNU Fortran can be downloaded and installed from source as described online. Note that Galacticus requires v4.9.0 or later of gcc-gfortran—earlier versions will not work.

**g++** Galacticus contains some C++ components. We use GNU C++ to compile Galacticus. You can use other C++ compilers, but we make no claims as to whether they will produce correctly working code. On Fedora the following (as root) will install GNU C++:

```
yum install gcc-c++
```

Alternatively, GNU C++ can be downloaded and installed from source as described online.

**GNU Scientific Library** The GNU Scientific Library (GSL) is used extensively by Galacticus to perform numerous numerical functions. We currently use v1.15 of GSL—earlier versions will not work (due to Galacticus’s use of the odeiv2 ODE solver interface that was introduced in GSL v1.15). On Fedora the following (as root) will install GSL:

```
yum install gsl gsl-devel
```

Alternatively, GSL can be downloaded and installed from source as described online.

**FoX** FoX is an XML parser for Fortran. The source can be downloaded from http://www1.gly.bris.ac.uk/~walker/FoX/source/FoX-4.1.1-full.tar.gz. We recommend following the instructions provided with the download for install. We use FoX v4.1.0.

**HDF5** The HDF5 specification is used for storing output data from Galacticus. We currently use HDF5\(^2\) v1.8.8. On Fedora the following (as root) will install the HDF5 libraries:

```
yum install hdf5 hdf5-devel
```

Alternatively, HDF5 can be downloaded and installed from source. If this is done, we recommend the following build sequence:

---

\(^1\) We cannot predict if significantly later versions of GNU Fortran will successfully compile Galacticus.

\(^2\) Early versions of HDF5 may work, but versions prior to v1.8.5 have known memory leak problems.
2.3. Required Libraries and Tools

F9X=gfortran
export F9X
./configure --prefix=/usr/local --enable-fortran --enable-production
make
make check
make install

FGSL  FGSL is a Fortran interface to the GNU Scientific Library and is used extensively by GALACTICUS. It can be downloaded from the link above. We use FGSL v0.9.4 and recommend the following build sequence.

./configure --f90 gfortran
make
make install

CPAN  GALACTICUS requires some Perl modules which probably are not installed by default. We recommend that you install these via CPAN which probably is installed by default. If it is not, on Fedora the following (as root) will install CPAN:

yum install perl-CPAN

Note that you can check if any of the following Perl modules are already installed using

perl -e "use Module::Name"

If no error message is given, the module is already installed. When installing modules using CPAN (see below for example), if the install fails because of a failed test, you can often force the install by running perl -MCPAN -e 'force("install","Module::Name")'. Of course, this may mean that the module is not working correctly...

Switch  This module implements distributed conditional testing. It is part of the core Perl distribution up to Perl v5.12.0. From Perl v5.14.0 onward it must be installed using

perl -MCPAN -e 'install Switch'

XML::Simple  Provides a simple interface in Perl for processing XML files. It is used extensively by GALACTICUS for building the code and for handling various data files. On Fedora the following (as root) will install XML::Simple:

yum install perl-XML-Simple

Alternatively, it can be installed from CPAN using:

perl -MCPAN -e 'install XML::Simple'

List::MoreUtils  Provides additional list-oriented utilities in Perl. It is used when automatically constructing lists of parameters that are accepted by a given executable when that executable is compiled. On Fedora the following (as root) will install List::MoreUtils:

yum install perl-List-MoreUtils
2. Installation

Alternatively, it can be installed from CPAN using:

```perl
perl -MCPAN -e 'install List::MoreUtils'
```

**List::Uniq** Provides functionality to find unique items in lists in Perl. It is used in GALACTICUS’s build system. It can be installed from CPAN using:

```perl
perl -MCPAN -e 'install List::Uniq'
```

**XML::SAX** Provides functionality to parse XML in Perl. It is used in GALACTICUS’s build system. On Fedora the following (as root) will install **XML::SAX**:

```bash
yum install perl-XML-SAX
```

Alternatively, it can be installed from CPAN using:

```perl
perl -MCPAN -e 'install XML::SAX'
```

**XML::Validator::Schema** Provides functionality to validate XML in Perl. It is used in GALACTICUS’s build system to validate the component domain specific language (DSL). It can be installed from CPAN using:

```perl
perl -MCPAN -e 'install XML::Validator::Schema'
```

**Sort::Topological** This module implements dependency based sort and is used by the GALACTICUS build system to ensure that tasks are performed in the correct order. It can be installed using

```perl
perl -MCPAN -e 'install Sort::Topological'
```

**Date::Time** This module implements handling of dates and times and is used by various scripts to timestamp files that they create. It can be installed using

```perl
perl -MCPAN -e 'install Date::Time'
```

**Data::Dumper** This module provides formatted output of arbitrary data structures. On Fedora the following (as root) will install **Date::Time**:

```bash
yum install perl-DateTime
```

Alternatively, it can be installed from CPAN using:

```perl
perl -MCPAN -e 'install Data::Dumper'
```

2.3.2. Typical Requirements

In addition to the tools listed in §2.3.1 a typical install (allowing you to run typical analysis tasks for example) requires the following tools to be installed:

**PDL::IO::HDF5** Provides a simple Perl interface to HDF5 files. It is used by analysis scripts to extract data from GALACTICUS output files. It can be installed from CPAN using:
2.3. Required Libraries and Tools

perl -MCPAN -e 'install PDL::IO::HDF5'

**poppler**  
**poppler** is a set of tools for working with PDF files. We use v0.22.2 and recommend the following build sequence.

```
./configure  
make  
make check  
make install
```

**IO::Compress::Bzip2**  This module provides the bzip2 compression tool in Perl. On Fedora the following (as root) will install **IO::Compress::Bzip2**:

```
yum install perl-Compress-Bzip2
```

Alternatively, it can be installed from CPAN using:

```
perl -MCPAN -e 'install IO::Compress::Bzip2'
```

**IO::Prompt**  This module provides convenient reading from the prompt in Perl. On Fedora the following (as root) will install **IO::Prompt**:

```
yum install perl-IO-prompt
```

Alternatively, it can be installed from CPAN using:

```
perl -MCPAN -e 'install IO::Prompt
```

**IO::Interactive**  This module provides functionality to determine if a Perl script is running interactively. It can be installed from CPAN using:

```
perl -MCPAN -e 'install IO::Interactive
```

**Text::Table**  This module provides formatted table output in Perl. On Fedora the following (as root) will install **Text::Table**:

```
yum install perl-Text-Table
```

Alternatively, it can be installed from CPAN using:

```
perl -MCPAN -e 'install Text::Table'
```

**Text::Wrap**  This module provides formatted paragraph output in Perl. It is most likely part of a standard Perl install, but if not it can be installed from CPAN using:

```
perl -MCPAN -e 'install Text::Wrap
```

**PDL**  Provides array math handling in Perl. It is used extensively by **GALACTICUS** for analysis of models. On Fedora the following (as root) will install **PDL**:

```
yum install perl-PDL
```
2. Installation

Alternatively, it can be installed from CPAN using:

```
perl -MCPAN -e 'install PDL'
```

**PDL::NiceSlice** Provides convenient array slicing for PDL. PDL::NiceSlice can be installed from CPAN using:

```
perl -MCPAN -e 'install PDL::NiceSlice'
```

**Math::SigFigs** Provides formatting of numbers to a given number of significant figures. It is used by GALACTICUS when making plots. It can be installed from CPAN using:

```
perl -MCPAN -e 'install Math::SigFigs'
```

**Astro::Cosmology** Provides basic cosmological calculations. It is used by GALACTICUS when making plots to convert data points measured under the assumptions of one set of cosmological parameters to the cosmological parameters assumed by GALACTICUS. It can be installed from CPAN using:

```
perl -MCPAN -e 'force("install","Astro::Cosmology")'
```

2.3.3. Full Requirements

In addition to the tools listed in §2.3.1 and §2.3.2 a full install (allowing you to run all scripts and programs included with GALACTICUS) requires the following tools to be installed:

**LaTeX::Encode** This module formats text for output to a \LaTeX\ document. It can be installed using:

```
perl -MCPAN -e 'install LaTeX::Encode'
```

**File::Find** This module provides functionality for searching directory structures for files. It can be installed from CPAN using:

```
perl -MCPAN -e 'install File::Find'
```

**File::Copy** This module provides interfaces for copying and moving files in Perl. It can be installed from CPAN using:

```
perl -MCPAN -e 'install File::Copy'
```

**Image::Magick** This module provides access to the IMAGEMAGICK tools from Perl. On Fedora the following (as root) will install Image::Magick:

```
yum install ImageMagick-perl
```

Alternatively, it can be installed from CPAN using:

```
perl -MCPAN -e 'install Image::Magick'
```

---

3On some systems, LaTeX::Encode seems to have problems. Once installed, try executing: perl -e "use LaTeX::Encode". If this issues any error messages, then you should locate the LaTeXEncode.pm file on your system edit it to comment out (or remove) the "use strict;" line.
2.3. Required Libraries and Tools

Term::ReadKey This module provides a simple interface for accepting key reads from standard input in
Perl. On Fedora the following (as root) will install Term::ReadKey:

```
yum install perl-Term::ReadKey
```
Alternatively, it can be installed from CPAN using:

```
perl -MCPAN -e 'install Term::ReadKey'
```

MIME::Lite This module provides simple e-mail sending functionality in Perl. On Fedora the following
(as root) will install MIME::Lite:

```
yum install perl-MIME-Lite
```
Alternatively, it can be installed from CPAN using:

```
perl -MCPAN -e 'install MIME::Lite'
```

Date::Time This module provides date/time formatting functionality in Perl. On Fedora the following
(as root) will install Date::Time:

```
yum install perl-Date::Time
```
Alternatively, it can be installed from CPAN using:

```
perl -MCPAN -e 'install Date::Time'
```

Net::SMTP::SSL This module implements an SSL authenticated SMTP e-mail protocol in Perl. On
Fedora the following (as root) will install Net::SMTP::SSL:

```
yum install perl-Net::SMTP::SSL
```
Alternatively, it can be installed from CPAN using:

```
perl -MCPAN -e 'install Net::SMTP::SSL'
```

Net::DBus This module implements interaction with the DBus message bus system in Perl. On Fedora
the following (as root) will install Net::DBus:

```
yum install perl-Net::DBus
```
Alternatively, it can be installed from CPAN using:

```
perl -MCPAN -e 'install Net::DBus'
```

PDL::MatrixOps Provides matrix operators for PDL. It is used when generating Monte Carlo samples
of parameters from covariance matrices. PDL::MatrixOps can be installed from CPAN using:

```
perl -MCPAN -e 'install PDL::MatrixOps'
```
2. Installation

**PDL::LinearAlgebra** Provides linear algebra algorithms for PDL. It is used for Cholesky decomposition when generating Monte Carlo samples of parameters from covariance matrices. **PDL::LinearAlgebra** can be installed from CPAN using:

```
perl -MCPAN -e 'install PDL::LinearAlgebra'
```

**Gnuplot** The **Gnuplot** plotting package is used extensively by **Galacticus** to produce plots. While not required for running **Galacticus** it is recommended as it allows you to easily check model results using the preexisting plotting scripts. On Fedora the following (as root) will install **Gnuplot**:

```
yum install gnuplot
```

Alternatively, **Gnuplot** can be installed from source, following the online instructions. Note that some scripts require **Gnuplot** v4.4 or later.

**GraphViz** The **GraphViz** package is used to create plots of merger tree structures. While not required to run **Galacticus** it is recommended to allow simple graphing of such structures (useful to get a visual impression of what’s going on in the code). On Fedora the following (as root) will install **GraphViz** and a module that allows Perl to interact with it:

```
yum install graphviz perl-GraphViz
```

Alternatively, **GraphViz** can be installed from source, following the online instructions, while the Perl module can be installed from CPAN using

```
perl -MCPAN -e 'install GraphViz'
```

### 2.4. Compiling Galacticus

To build **Galacticus** (after installing all required libraries) ensure that you are in the **Galacticus/v0.0.1** directory and then simply type:

```
make Galacticus.exe
```

This will create the **Galacticus.exe** executable. The build takes some time, you’ll see a set of XML files get created first which **Galacticus** uses to figure out how modules link in to the **Galacticus** code. After that, the Fortran files are compiled. We regularly build **Galacticus** using a parallel make without any problems.

The **Makefile** contains a few options that you may want to adjust:

**FCCOMPILER** The command to invoke a Fortran 2003 compliant compiler. **Galacticus** should compile with any such compiler. In practice, we have only tried it using **GFortran v4.9.0+**. In particular, **Galacticus** makes use of certain Fortran 2003 features (notably procedure pointers, type-bound procedures and class variables) which older compilers might not handle (and some newer compilers might still have difficulty with);

**CCOMPILER** The command to invoke a C compiler.

**CPPCOMPILER** The command to invoke a C++ compiler.

---

4We cannot predict if significantly later versions of GNU Fortran will successfully compile **Galacticus**.
2.4. Compiling Galacticus

**PREPROCESSOR** The command to invoke a preprocessor to handle #IFDEF etc. statements. We normally use cpp.

**MODULETYPE** A label identifying the structure of modules built by the compiler. This is used to detected when module interfaces have been changed (requiring a recompile of all dependent code) and when only the module internals have changed. When the GFORTRAN compiler is used GCC-f95-on-LINUX should be used here. Look in ./scripts/build/Compare_Module_Files.pl for other possibilities (if your compiler isn’t listed you’ll need to either edit that script or deal with longer recompiles if you edit the code).

**FCFLAGS** Flags to pass to the Fortran compiler. Standard options for both error-checking and optimized builds for the GFORTRAN compiler are given in the Makefile.

**CFLAGS** Flags to pass to the C compiler.

**CPPFLAGS** Flags to pass to the C++ compiler.

Additionally, you can add compiler options to the GALACTICUS_FCFLAGS, GALACTICUS_CFLAGS, and GALACTICUS_-CPPFLAGS environment variables. This is useful to add machine-specific options.

### 2.4.1. Compiling with OpenMP Parallelism

By default, GALACTICUS will be compiled to run in parallel on machine with multiple CPUs using OpenMP. To disable this simply remove (or comment out the)

**FCFLAGS += -fopenmp**

line in the Makefile. When running GALACTICUS in parallel using OpenMP it may be necessary to increase the stack size allocated to each thread (since GALACTICUS calls some procedures recursively this can result in large numbers of local variables being allocated on the stack). To do this, use

```
setenv KMP_STACKSIZE 16777216
```

in csh and variants or

```
export KMP_STACKSIZE=16777216
```

in bash and variants. If you get stack overflows while running GALACTICUS in parallel, try increasing this value further.

GALACTICUS currently implements parallel calculations at the tree level—that is, each parallel thread works on a separate merger tree. This means that you need sufficient memory to hold multiple merger trees at once. The advantage of this approach is that it is highly scalable (assuming no need for communication between trees), and GALACTICUS can achieve close to optimal speed-up in many cases. The limit to the speed-up is usually determined by the workload balance between the trees. If one tree requires significantly more time to run that all other trees combined then one thread will be left working on that tree while all others have finished. To limit this problem, it is recommended that parallel runs which use `mergerTreeConstructMethod=build` (see §sec:MergerTreeConstruction) be conducted with the `[mergerTreeBuildTreesProcessDescending]=true` parameter set to true. This will cause the most massive merger trees (i.e. those which take longest to process) to be processed first.
2. Installation

2.5. Installing Without Root Access

It is possible to install GALACTICUS without root access to your computer. The following approach has worked in many cases—some adjustments may be required for your specific system. Depending on what is already installed on your system, you may be able to skip some of the following installs. Refer to §2.3.1, §2.3.2 and §2.3.3 to decide which of the following tools you want to install. (Or, alternatively, you may need to install additional tools.) Choose a location to install that has at least 4Gb of free space. In the following, this install location is referred to as /your/install/path. In the following, we assume you’re using some variant of the C-shell. If you’re using bash or some other Bourne shell, setenv VAR abcd should be translated to export VAR=abcd.

! Install GFortran, GCC and G++:

```
svn co svn://gcc.gnu.org/svn/gcc/trunk gcc-trunk
cd gcc-trunk
svn up
cd ..
rm -rf gcc-build
mkdir gcc-build
cd gcc-build
./gcc-trunk/configure --prefix=/your/install/path --enable-languages=c,c++,fortran --disable-multilib
make
make install
```

! Add to .cshrc (or equivalent):

```
setenv PATH /your/install/path/bin:$PATH
setenv LD_LIBRARY_PATH /your/install/path/lib:$LD_LIBRARY_PATH
```

! Install GSL:

```
wget "http://www.mirrorservice.org/sites/ftp.gnu.org.gnu/gsl/gsl-1.15.tar.gz"
tar xvfz gsl-1.15.tar.gz
cd gsl-1.15
./configure --prefix=/your/install/path
make
make check
make install
```

! Install FGSL:

```
wget "http://www.lrz-muenchen.de/services/software/mathematik/gsl/fortran/fgsl-0.9.4.tar.gz"
tar xvfz fgsl-0.9.4.tar.gz
cd fgsl-0.9.4
./configure --f90 gfortran --gsl /your/install/path
make
make install
```
2.5. Installing Without Root Access

! Install zlib:

```bash
wget "http://zlib.net/zlib-1.2.5.tar.gz"
tar xvfz zlib-1.2.5.tar.gz
cd zlib
./configure --prefix=/your/install/path
make
make check
make install
```

! Install HDF5:

```bash
wget http://www.hdfgroup.org/ftp/HDF5/current/src/hdf5-1.8.7.tar.gz
tar xvfz hdf5-1.8.7.tar.gz
cd hdf5-1.8.7
setenv F9X gfortran
./configure --prefix=/your/install/path --enable-fortran --enable-production
--with-zlib=/your/install/path
make
make check
make install
```

! Install FoX:

```bash
wget "http://www1.gly.bris.ac.uk/~walker/FoX/source/FoX-4.1.0-full.tar.gz"
tar xvfz FoX-4.1.0-full.tar.gz
cd FoX-4.1.0
setenv FC gfortran
./configure --prefix=/your/install/path
make
make check
make install
```

! Install Mercurial:

```bash
wget "http://mercurial.selenic.com/release/mercurial-2.4.1.tar.gz"
tar xvfz mercurial-2.4.1.tar.gz
cd mercurial-2.4.1
setenv PREFIX /your/install/path
make
make install
```

! Install Poppler:

```bash
wget "http://poppler.freedesktop.org/poppler-0.22.2.tar.gz"
tar xvfz poppler-0.22.2.tar.gz
cd poppler-0.22.2
./configure --prefix=/your/install/path
make
```
2. Installation

make check
make install

! Install Perl local::lib for local installs of modules:

mkdir .cpan
mkdir perl5
ln −sf /your/install/path/.cpan $HOME/
ln −sf /your/install/path/perl5 $HOME/
wget http://search.cpan.org/CPAN/authors/id/A/AP/APEIRON/local-lib−1.008004.tar.gz
tar xvfz local−lib−1.008004.tar.gz
cd local−lib−1.008004
perl Makefile.PL --bootstrap
make
make test
make install
perl −I$HOME/perl5/lib/perl5 −Mlocal::lib >> $HOME/.cshrc

! Install Perl modules:

perl −MCpan −e "install Sort::Topological"
perl −MCpan −e "install LaTeX::Encode"
perl −MCpan −e "install XML::Simple"
perl −MCpan −e "install Math::SigFigs"
perl −MCpan −e "install GraphViz"
perl −MCpan −e 'force("install","Astro::Cosmology")'
perl −MCpan −e "install File::Find"
perl −MCpan −e "install File::Copy"
perl −MCpan −e "install Image::Magick"
perl −MCpan −e "install Term::ReadKey"
perl −MCpan −e "install MIME::Lite"
perl −MCpan −e 'install Text::Table'
perl −MCpan −e 'install IO::Compress::Bzip2'
perl −MCpan −e 'install Date::Format'
perl −MCpan −e 'install Net::SMTP::SSL'
perl −MCpan −e 'install Net::DBus'
perl −MCpan −e 'install PDL'
perl −MCpan −e 'install PDL::LinearAlgebra'
perl −MCpan −e 'install PDL::MatrixOps'
perl −MCpan −e 'install PDL::NiceSlice'
perl −MCpan −e 'install PDL::IO::HDF5'

! Install GnuPlot:

wget "http://downloads.sourceforge.net/project/gnuplot/gnuplot/4.4.3/gnuplot−4.4.3.tar.gz"
tar xvfz gnuplot−4.4.3.tar.gz
cd gnuplot−4.4.3
./configure --prefix=/your/install/path
make
make install

! Install GraphViz:

tar xvfz graphviz-2.28.0.tar.gz
cd graphviz-2.28.0
./configure --prefix=/your/install/path
make
make check
make install

! Install Galacticus:

wget "http://users.obs.carnegiescience.edu/abenson/galacticus/versions/galacticus_v0.9.3.tar.bz2"
tar xvfj galacticus_v0.9.3.tar.bz2
cd Galacticus/v0.9.3

! Add to Makefile (below the GALACTICUS_FCFLAGS = line):

GALACTICUS_FCFLAGS += -fintrinsic-modules-path /your/install/path/finclude
 -fintrinsic-modules-path /your/install/path/include/gfortran
 -fintrinsic-modules-path /your/install/path
 -fintrinsic-modules-path /your/install/path/include
 -L /your/install/path/lib

It should then be possible to compile and run GALACTICUS.

2.6. Installing on Mac OS X

The following guidelines have been tested on a MacBook Pro, running Mac OS X V10.6.8.

2.6.1. Update GNU compilers

It is likely that the default compiler is older than GCC v4.9 that is required to properly compile the code without any errors. Use a package manager to download as recent a version as possible of gcc. For the case of MacPorts, this requires,

$ sudo port -v install gcc48 +gfortran

and reset the default version of gcc compilers by first listing available options

$ sudo port select --list gcc

and then explicitly setting to mp-gcc48 by

$ sudo port select --set mp-gcc48
2.6.2. Installing HDF5, FoX and FGSL

Once you have installed the latest compiler suite, you will need to recompile your HDF5, FoX and FGSL libraries. Download FoX (for XML parsing) from

http://www1.gly.bris.ac.uk/~walker/FoX/source/FoX-4.1.2-full.tar.gz

and run configure

$ sudo ./configure --prefix=/opt/local

i.e. install libraries in the same branch as the MacPorts distribution. Then do the usual

$ sudo make clean; sudo make; sudo make check; sudo make install

Similarly download FGSL from

http://www.lrz.de/services/software/mathematik/gsl/fortran/

and, assuming that you have downloaded GSL with MacPorts and they are installed in /opt/local/include and /opt/local/lib, run configure

$ sudo ./configure --f90 gfortran --gsl /opt/local --prefix /opt/local

i.e. install libraries in the same branch as the MacPorts distribution. Then do the usual

$ sudo make clean; sudo make; sudo make install

Finally download the latest version of HDF5, configure

$ sudo ./configure --enable-fortran --prefix=/opt/local

and do the usual

$ sudo make clean; sudo make; sudo make test; sudo make install

This should ensure that the modules (hdf5.mod, fox_dom.mod, ...) are compatible with the Galacticus build.

Galacticus requires crypt.h to install on linux-based systems; this is part of the GNU C library glibc. However, glibc has not been ported to Mac OS X and so this will not work properly. To get it working, edit source/utility.hashes.cryptographic.md5.c and comment out #include <crypt.h>, replacing it with #include <unistd.h>.

Note that you will need to amend the Makefile so that Galacticus knows where these .mod files are, which you can do by adding

FCFLAGS += -fintrinsic-modules-path /opt/local/include
-fintrinsic-modules-path /opt/local/include/gfortran
-fintrinsic-modules-path /opt/local/finclude.

You can also add

export GALACTICUS_FCFLAGS = "-L/opt/local/lib"

You can also add

to your .profile (i.e. .bashrc) file so that Galacticus knows where to find libraries (for FoX, libcrypt, ...) during linking.
2.6.3. Installing Perl Modules

Follow the instructions in the previous section to download and install new perl modules. You may need to upgrade CPAN using

\$ sudo perl -MCPAN -e 'install Bundles::CPAN'

before installation of certain modules (e.g. DateTime.pm) would proceed correctly. To install PDF::Labels, I found it necessary to

\$ sudo perl -MCPAN -e 'install PDF::Create''

and then

\$ sudo perl -MCPAN -e 'install PDF::Labels''
3. Running Galacticus

3.1. Configuration File

The file `galacticusConfig.xml`, is present, is used to configure GALACTICUS and provide useful information. It should have the following structure:

```xml
<config>
  <contact>
    <name>My Name</name>
    <email>me@ivory.towers.edu</email>
  </contact>
  <email>
    <host>
      <name>myComputerHostName</name>
      <method>smtp</method>
      <host>smtp-server.ivory.towers.edu</host>
      <user>myUserName</user>
      <passwordFrom>kdewallet</passwordFrom>
    </host>
    <host>
      <name>default</name>
      <method>sendmail</method>
    </host>
  </email>
</config>
```

The name and e-mail address in the `contact` section will be stored in any GALACTICUS models run—this helps track the provenance of the model. The `email` section determines how e-mail will be sent. Within this section, you can place one or more `host` elements, the `name` element of which specifies the host name of the computer to which these rules apply (the `default` host is used if no other match is found). For each host, the `method` element specifies how e-mail should be sent, either by `sendmail` or via `smtp`. For SMTP transport (which currently supports SSL connections only), you must specify the `host` SMTP server, `user` name. The `passwordFrom` element specifies how the password for the SMTP log in should be obtained. If set to `input` then the user will be prompted for the password as needed. Alternatively, if you use the KDE desktop and the KDEWallet password manager, setting `passwordFrom` to `kdewallet` will cause the password to be stored in the KDE wallet and retrieved from there subsequently.

3.2. Parameter Files

GALACTICUS requires a file of parameters to be given as a command line argument. The parameter file is an XML file (which makes it easy to manipulate and construct these files from within many languages, e.g. Perl) with the following structure:

```xml
<parameters>
```
3. Running Galacticus

```xml
<parameter>
  <name>parameter1name</name>
  <value>parameter1value</value>
</parameter>
```

Each parameter element contains name and value elements which contain the parameter name and desired value respectively. The value can be a number, word(s) or an array of space-separated numbers or words. Parameters are used to control the values of numerical parameters and also to select methods and other options. If a parameter is not specified in the file a default value (hard coded into GALACTICUS) will be used instead. The default values have been chosen to produce a realistic model of galaxy formation, but may change as GALACTICUS evolves.

All parameter values (both those specified in this file and those set to default) used during a GALACTICUS run are output to the Parameters group within the GALACTICUS output file. The script scripts/aux/Extract_Parameter_File.pl will, if given a GALACTICUS output file, extract the parameters from it and output them into an XML file suitable for re-input into GALACTICUS. If parameters are present in the parameter file which do not match any known parameter in GALACTICUS then a warning message, listing all unknown parameters, will be given when GALACTICUS is run. Note that this will not prevent GALACTICUS from running—sometimes it is convenient to include parameters which are not used by GALACTICUS, but which might be used by some other code.

3.2.1. Validating Parameter Files

A script, scripts/aux/validateParameters.pl, is provided to validate parameter files and thereby ensure that they are consistent with GALACTICUS's expectations and requirements. To use simply execute:

```
scripts/aux/validateParameters.pl myParameters.xml
```

No output (and an exit value of 0) indicates a valid parameter file. Invalid parameter files will result in an exit value other than 0 and will produce error messages that should help to track down the problem with the file.

3.2.2. Generating Parameter Files

Some scripts are provided which assist in the generation of parameter files. These are located in the scripts/parameters/ folder and are detailed below:

```bash
cosmologicalParametersMonteCarlo.pl
```

This script will generate a set of cosmological parameters drawn at random from the WMAP-9 constraints Hinshaw et al. [2012]. It uses the covariance matrix (currently defined in data/Cosmological_Parameters_WMAP-9.xml) to produce correlated random variables\(^1\). The generated parameters are printed to standard output as GALACTICUS-compatible XML.

---

\(^1\)Note that this does not capture the full details of the correlations between parameters, since it uses just the covariance matrix. For a more accurate calculation the full Monte Carlo Markov Chains used in the WMAP-9 parameter fitting should be used instead.
3.3. Running Galacticus

**Galacticus** is running using

```
Galacticus.exe [<parameterFile>]
```

where `parameterFile` is the name of the file containing a list of parameter values for **Galacticus**. **Galacticus** will display messages indicating its progress as it runs (the verbosity can be controlled with the `verbosityLevel` parameter). Usually, the **Galacticus** executable should be invoked from the directory in which it was built. However, you can choose to set the environment variable `GALACTICUS_ROOT_V091` to the full path to the build directory, in which case the **Galacticus** executable can be invoked from anywhere and will access all required files and scripts relative to this path. This can allow multiple users to all make use of the same **Galacticus** install.

3.3.1. Writing Data To a Temporary File

When running **Galacticus** on a compute cluster it is often advantageous to have output written to a local scratch disk during run time and only moved to networked storage after the run is complete. (Otherwise, **Galacticus** will perform many small writes to networked storage which can result in extremely slow run times.) To do this, simply set the parameter `[galacticusOutputScratchFileName]` to the full path of a file to write to on local scratch space. During the run, data will be written to this file. After the run is finished, **Galacticus** will move this file to its permanent location as specified by the parameter `[galacticusOutputFileName]`.

3.3.2. Restarting A Crashed Run

If **Galacticus** crashes, it can be useful to restart the calculation from just prior to the crash to speed the debugging process. **Galacticus** has functionality to store and retrieve the internal state of any modules and to recover this to permit such restarting. Currently, this is implemented with the build and read methods of merger tree construction, such that the internal state is stored prior to commencing the building or reading of each tree, thereby allowing a calculation to be restarted with the tree that crashed. More general store/retrieve behavior is planned for future releases.

To cause **Galacticus** to periodically store its internal state include the following input parameter:

```xml
<parameter>
  <name>stateFileRoot</name>
  <value>galacticusState</value>
</parameter>
```

This will cause the internal state to be stored to files `galacticusState.state` and `galacticusState.fgsl.state` prior to commencing building each merger tree. Should a tree crash then replace this input parameter with:

```xml
<parameter>
  <name>stateRetrieveFileRoot</name>
  <value>galacticusState</value>
</parameter>
<parameter>
  <name>mergerTreeBuildTreesBeginAtTree</name>
  <value>N</value>
</parameter>
```
3. Running Galacticus

where \( N \) is the number of the tree that crashed. This will cause calculations to begin with tree \( N \) and for the internal state to be recovered from the above mentioned files. The resulting tree and all galaxy formation calculations should therefore proceed just as in the original run (and so create the same crash condition).

**OpenMP**

When running a model in parallel using OpenMP, a separate state file will be written for each thread, with the thread number appended to the end of each state file name. For debugging purposes, it is suggested that a crashed OpenMP run be restarted using just a single thread. To do this, change the appended thread number on the state files corresponding to the thread which crashed to 0 such that they will be used by the single thread when the run is restarted.

3.3.3. Running Grids of Models

You can easily write your own scripts to generate parameter files and run GALACTICUS on these files. An example of such a script is `scripts/aux/launch.pl`. This script will loop over a sequence of parameter values, generate appropriate parameter files, run GALACTICUS using those parameters and analyze the results. This script currently supports running of GALACTICUS on a local machine, via a PBS queue (as multiple jobs or a single job), or on a CONDOR cluster. To run the script simply enter:

```
./scripts/aux/launch.pl <runFile>
```

This will launch a single instance of the script. Multiple instances can be launched and will share the work load (i.e. they will not attempt to run a model which another instance is already running or has finished). If multiple instances are to be launched on multiple machines a command line option to `launch.pl` can be used to ensure that they do not duplicate work. Adding `-instance 2:4` for example will tell the script to run only the second model from each block of four models it finds. Launching for `launch.pl` scripts on four different machines with `-instance 1:4`, `-instance 2:4`, `-instance 3:4` and `-instance 4:4` will then divide the models between those machines.

The `runFile` is an XML file with the following structure:

```xml
<parameterGrid>
  <modelRootDirectory>models.new</modelRootDirectory>
  <baseParameters>newBestParametersQuick.xml</baseParameters>
  <compressModels>no</compressModels>
  <splitModels>4</splitModels>

  <launchMethod>pbs</launchMethod>

  <local>
    <threadCount>3</threadCount>
    <ompThreads>4</ompThreads>
  </local>

  <condor>
    <galacticusDirectory>/home/condor/Galacticus/v0.9.3</galacticusDirectory>
    <universe>vanilla</universe>
    <environment>LD_LIBRARY_PATH=/usr/lib:/usr/lib64:/usr/local/lib</environment>
    <requirement>Memory &gt;= 1000 &amp;&amp; Memory &lt; 2000</requirement>
    <transferFile>{PWD}/myFile.data</transferFile>
  </condor>
</parameterGrid>
```
3.3. Running Galacticus

```xml
<wholeMachine>true</wholeMachine>
<postSubmitSleepDuration>5</postSubmitSleepDuration>
<jobWaitSleepDuration>10</jobWaitSleepDuration>
</condor>

<pbs>
<scratchPath>/scratch/me</scratchPath>
<wallTime>48:00:00</wallTime>
<memory>3gb</memory>
<ompThreads>8</ompThreads>
<queue>standard</queue>
<maxJobsInQueue>10</maxJobsInQueue>
<mpiLaunch>yes</mpiLaunch>
<mpiRun>/opt/openmpi/bin/mpirun</mpiRun>
<environment>LD_LIBRARY_PATH=/home/me/software/Galacticus/Tools/lib64:$LD_LIBRARY_PATH</environment>
<postSubmitSleepDuration>10</postSubmitSleepDuration>
<jobWaitSleepDuration>60</jobWaitSleepDuration>
</pbs>

<monolithicPBS>
<mpiLaunch>yes</mpiLaunch>
<nodes>1</nodes>
<threadsPerNode>12</threadsPerNode>
<ompThreads>6</ompThreads>
<jobWaitSleepDuration>60</jobWaitSleepDuration>
<analyze>no</analyze>
<environment>LD_LIBRARY_PATH=/home/me/software/Galacticus/Tools/lib64:$LD_LIBRARY_PATH</environment>
<includePath>/my/include/path</includePath>
<libraryPath>/opt/sgi/mpt/mpt-2.04/lib</libraryPath>
<shell>csh</shell>
<pbsCommand>source /usr/share/modules/init/csh</pbsCommand>
<pbsCommand>module load mpi-sgi/2.04_64</pbsCommand>
</monolithicPBS>

<parameters>
<label>modelLabel</label>
<parameter>
    <name>stabilityThresholdStellar</name>
    <value>1.1</value>
    <value>0.9</value>
</parameter>
</parameters>

<parameters>
<parameter>
    <name>stabilityThresholdGaseous</name>
    <value>1.1</value>
    <value>0.9</value>
</parameter>
</parameters>
```
Each parameters block contains a list of parameters following the format used in standard GALACTICUS parameter files, with the difference that each parameter can have multiple values. A model will be run for all possible combinations of these values. Additionally, any value element may contain further parameter elements. All possible values of these parameters will be looped over when, and only when, the appropriate value of the containing parameter is being used. For example, in the above example, models will be run with \([\text{starFormationKennicuttSchmidtTruncate}] = \text{true} \) and \(\text{false}\) only when \([\text{starFormationTimescaleDisksMethod}] = \text{Kennicutt-Schmidt}\) and not when \([\text{starFormationTimescaleDisksMethod}] = \text{dynamicalTime}\).

It is also possible to set the value of a parameter by modifying the current value using a Perl regular expression. This is done by given a modify element instead of a value element in the parameter definition. The modify element must contain find and replace elements, the first of which must be a valid Perl regular expression, and the second of which must specify the replacement text. In the above example, the parameters block specifies that models are two be run for \([\text{imfSelectionFixed}] = \text{Salpeter}\) and \(\text{Chabrier}\). However, for the case of \([\text{starFormationKennicuttSchmidtTruncate}] = \text{false}\) these parameters will be modified by suffixing them with “Truncated”.

Some variables, which are expanded at run time, are available. These include:

- %%galacticusOutputPath%% This will be expanded to the output path of a model. Useful for specifying paths for any additional output.

By default, each model is output into a sequentially numbered directory within the ./models directory.
By default, these directories have the prefix *galacticus*. This can be changed by including a *label* element inside a *parameters* block, in which case the content of the *label* element will be used as the prefix. This root directory can be modified by the optional *modelRootDirectory* element. Additionally, a set of base parameters can be read from a file specified by the *baseParameters* file—these will be read before each model is run and before any variations in parameters for the specific model are applied. As such, it defines the default model around which parameter variations occur. Additional options that may be present in the file (as elements within the *parameterGrid* element) are:

- **doAnalysis** If set to “no” then no analysis scripts will be run on completed models, otherwise, they will be. Optionally, the analysis script to run can be specified via the *analysisScript* element (see §3.3.4);
- **emailReport** If set to “yes” a report will be e-mailed to the address specified in *galacticusOptions.xml* when a model fails. Otherwise, the report will be written to standard output instead.
- **compressModels** If “no” then models are not compressed after being run. Otherwise, the contents of the model output directory will be compressed using *bzip2*.
- **splitModels** If set to an integer larger than 1, each *Galacticus* model will be split into that number of jobs, and those jobs will be launched (using the selected method) independently. Once finished, the outputs from these split models will be merged back into a single model. This allows, for example, effectively distributing a single *Galacticus* model over multiple nodes of a PBS cluster.

The method by which to launch jobs must be specified in the *launchMethod* element. Currently available options are:

- **local** The models will be run on the local machine. Two additional options can be specified within a *local* XML block:
  - **threadCount** The number of individual model threads to be launched.
  - **ompThreads** The number of OpenMP threads to be used by each model.
- **pbs** Jobs will be submitted to a PBS batch queue system. The following options are available and can be specified within a *pbs* XML block:
  - **scratchPath** An optional path to which the model output will be written at run time. At the completion of each run, the data will be transferred to the usual output location. This is useful to avoid network I/O during run time;
  - **wallTime** A limit on the wall time allowed for each model (optional);
  - **memory** A limit on the memory allowed for each model (optional);
  - **ompThreads** The number of OpenMP threads to use for each model (optional). This is used to request an appropriate number of processors per node;
  - **queue** The name of the queue to submit the jobs to (optional);
  - **maxJobsInQueue** The maximum number of jobs to place in the queue. Additional jobs will be held and submitted once the number of jobs in the queue drops below this value (optional);
- **mpiLaunch** If set to “yes” then the *mpirun* command will be used to launch a single copy of *Galacticus* (which may then spawn multiple OpenMP threads). If instead set to “no” then *Galacticus* is launch without the use of the *mpirun* command. Some systems will limit a code launched with *mpirun* to using just a single CPU (even if multiple OpenMP threads are spawned). In such cases, setting this option to “no” should permit multiple CPUs to be utilized.
3. Running Galacticus

mpiRun The path to the mpirun executable (optional—if not present, mpirun must be in PATH);

environment Any settings here are set in each PBS job in order to set appropriate environment variables on the machine where a job is executed;

analyze If set to “yes” then analysis (if any) will be performed as part of the PBS job. Otherwise, analysis is performed by the submitting machine.

postSubmitSleepDuration The time (in seconds) to wait after submitting each job. This prevents flooding the PBS queue manager with a large number of jobs in rapid succession.

jobWaitSleepDuration The time (in seconds) to sleep between successive checks of the PBS queue to see if any of the submitted jobs have finished.

monolithicPBS A single job will be submitted to a PBS batch queue system. This job will internally run multiple copies of GALACTICUS each with a different set of parameters. The following options are available and can be specified within a monolithicPBS XML block:

nodes The total number of nodes to use for the PBS job.

threadsPerNode The number of threads per node to use for the PBS job.

ompThreads The number of OpenMP threads to use for each model (optional). This is used to request an appropriate number of processors per node, and must be a factor of threadsPerNode;

scratchPath An optional path to which the model output will be written at run time. At the completion of each run, the data will be transferred to the usual output location. This is useful to avoid network I/O during run time;

wallTime A limit on the wall time allowed for each model (optional);

memory A limit on the memory allowed for each model (optional);

queue The name of the queue to submit the jobs to (optional);

mpiRun The path to the mpirun executable (optional—if not present, mpirun must be in PATH);

environment Any settings here are set in each PBS job in order to set appropriate environment variables on the machine where a job is executed;

analyze If set to “yes” then analysis (if any) will be performed as part of the PBS job. Otherwise, analysis is performed by the submitting machine.

postSubmitSleepDuration The time (in seconds) to wait after submitting each job. This prevents flooding the PBS queue manager with a large number of jobs in rapid succession.

jobWaitSleepDuration The time (in seconds) to sleep between successive checks of the PBS queue to see if any of the submitted jobs have finished.

condor Jobs will be submitted to a Condor cluster. The following options are available and can be specified within a condor XML block:

galacticusDirectory When a GALACTICUS job is submitted to a CONDOR cluster the GALACTICUS executable and the input parameter file are transferred to the machine where the job runs. Other files, such as data files, are not transferred. Therefore, they must be already present on any remote machine on which the job can run. This option specifies where a complete GALACTICUS installation can be found on the remote machine. If not present, it defaults to /home/condor/Galacticus/v0.9.0;

universe Specifies to which CONDOR universe jobs should be submitted. Allowed options are “vanilla” and “standard”. If the standard universe is to be used then GALACTICUS must have been linked with condor_compile—the Makefile allows this if the relevant lines are uncommented;
3.3. Running Galacticus

**environment** Any settings here are passed to CONDOR’s `environment` option in order to set appropriate environment variables on the machine where a job is executed;

**requirement** Any setting here is passed to CONDOR’s `requirements` option to specify requirements for each job. Multiple `requirement` entries will be combined (using logical and).

**transferFile** Any files listed here will be transferred the Condor worker (and so will be accessible from the path in which GALACTICUS is running). The macro `{PWD}` will be automatically expanded to the present working directory. Multiple `transferFile` entries can be given.

**wholeFile** Setting this option to `true` will add `+RequiresWholeMachine = True` to the Condor submit file. If Condor has been configured to allow jobs to take over a whole machine\(^2\), this will cause jobs to do so. This is useful if you want to run OpenMP GALACTICUS on a Condor cluster.

**postSubmitSleepDuration** The time (in seconds) to wait after submitting each job. This prevents flooding the Condor queue manager with a large number of jobs in rapid succession.

**jobWaitSleepDuration** The time (in seconds) to sleep between successive checks of the Condor queue to see if any of the submitted jobs have finished.

In addition to the `galacticus.hdf5` output file, each model directory will contain a file `newParameters.xml` which contains the parameters used to run the model and `galacticus.log` which contains any output from GALACTICUS during the run.

If present, the file `galacticusConfig.xml`, described in §3.1, is parsed for configuration options. If the contact element is present, the listed name and e-mail address will be used to determine who should receive error reports should a model crash. The error report will contain the host name of the computer running the model, the location of the model output and the log file (which may be incomplete if output is being buffered). Additionally, any core file produced will be stored in the model directory for later perusal, and the state files (see §3.3.2) for the run can also be found in the model directory.

3.3.4. Analysis of Models

The `Run_Galacticus.pl` script will automatically run `scripts/analysis/Galacticus_Compute_Fit.pl` on each model to generate plots and fitting data unless `doAnalysis=no` is set in the `runFile` (see §3.3.3). This script, which can also be running manually using

```
./scripts/analysis/Galacticus_Compute_Fit.pl <galacticusFile> <outputDirectory> [<analysisFile>]
```

where `galacticusFile` is the name of the GALACTICUS output file to analyze and `outputDirectory` is the directory into which plots and fitting data should be placed, reads the file `<analysisFile>` (or `data/Galacticus_Compute_Fit_Analyses.xml` if `<analysisFile>` is not specified) which has the following structure:

```
<analyses>
  <analysis>
    <script>scripts/plotting/Plot_HI_Mass_Function.pl</script>
    <weight>1.0</weight>
  </analysis>
  <analysis>
    <script>scripts/plotting/Plot_K_Luminosity_Function.pl</script>
    <weight>1.0</weight>
  </analysis>
</analyses>
```

\(^2\)As described here for example.
3. Running Galacticus

Each analysis element contains the name of a script to run to perform some analysis and a weight to be given to the results of this analysis when combining results to get a net goodness of fit. Each script listed will be run and is expected to have accept arguments of the form:

```
My_Analysis_Script.pl <galacticusFile> <outputDirectory> <showFit>
```

where the showFit argument can be 0 or 1 and, if set to 1, the script should output an XML chunk to standard output giving details of its fitting analysis. This chunk should have the form:

```
<galacticusFit>
  <name>Description of this analysis</name>
  <chiSquared>24.5</chiSquared>
  <degreesOfFreedom>19</degreesOfFreedom>
  <fileName>Output_File_Name.pdf</fileName>
</galacticusFit>
```

where chiSquared and degreesOfFreedom are the fitting results. All such data returned from fitting scripts will be collated by Galacticus_Compute_Fit.pl, augmented with the weight value and the net goodness of fit determined. All of this information is then output to galacticusFits.xml in the selected output directory.

Performing Other Analysis

If `<doAnalysis>` = yes and `<analysisFile>` is set to something other than an XML file it is assumed that this is an analysis script that should be run directly. The script will be executed with the output directory for the Galacticus model as the first and only argument.

3.3.5. Running Models in “Embarrassingly Parallel” Mode

While Galacticus is parallelized via OpenMP it is also possible to split a given model across several “worker” CPUs on one or more computers. The trees to be processed will be shared between these workers and the results can be later recombined. To use this “poor man’s” parallelization, add the following to a model parameter file:

```
<parameter>
  <name>treeEvolveWorkerCount</name>
  <value>N</value>
</parameter>
<parameter>
  <name>treeEvolveWorkerNumber</name>
  <value>i</value>
</parameter>
```

where N is the total number of workers to be used and i is the number of this worker (ranging from 1 to N). You can generate these individual input parameter files from a single base parameter file using:

```
scripts/aux/Split_Models_For_Workers.pl <parameterFile> <workerCount>
```
where <parameterFile> is the name of the base parameter file and <workerCount> is the number of workers required. The script will create an input file for each worker (input files will have the same name as the base parameter file but with a “_N”, where N is the worker number, inserted before the “.xml”). Output file name for each worker will be the same as specified in the base parameter file, but with a “_N”, where N is the worker number, inserted before the “.xml”.

Once all workers have finished, their outputs can (if required) be combined into a single output file using the Merge_Models.pl script as follows:

```
./scripts/aux/Merge_Models.pl <model1> <model2> .... <modelOutput>
```

where model1 etc. are the names of the various output files and modelOutput is the file into which the combined results should be placed. The Merge_Models.pl script will combine all merger trees into the output file and will additionally cumulate any data in the globalHistory groups in these files. The UUIDs of the merged files (see §5.1.1) will be concatenated (with a “:” separator) and placed into the UUIDs attribute of the new file. Additionally, a new UUID will be generated and stored in the UUID attribute of the new file.

### 3.3.6. Limiting the Load Average

If [treeEvolveLimitLoadAverage]=true then GALACTICUS will attempt to keep the load average of the system under [treeEvolveLoadAverageMaximum] by waiting to run trees if the current load average exceeds this value. [treeEvolveLoadAverageMaximum] can be set to the numerical maximum load average desired or, alternatively, can be set to processorCount in which case the number of processor cores present on the system will be used for [treeEvolveLoadAverageMaximum].

### 3.4. Additional Codes

The GALACTICUS code base can be used for other calculations. Some examples of such usage (and which are sufficiently useful in their own right) are included and are detailed in this section.

#### 3.4.1. Excursion_Sets

The Excursion_Sets code will generate an HDF5 output file which contains a variety of measures related to excursion sets in the Press-Schechter formalism. The code is built and run as follows:

```
make Excursion_Sets.exe
Excursion_Sets.exe <parameterFile> <outputFile>
```

where parameterFile is a file of parameters in GALACTICUS’s usual XML format and outputFile is the name of the file to which the excursion set data will be written. The output file has the following structure:

```
+-> barrier [dataset]
|  +-> firstCrossingProbability [dataset]
|  +-> firstCrossingRate [dataset]
|  +-> haloMass [dataset]
|  +-> haloMassFunction [dataset]
```
### 3. Running Galacticus

|  
| +-> powerSpectrum [dataset]  
|   
| +-> variance [dataset]  
|   
| +-> wavenumber [dataset]  

These datasets contain the following information:

- **haloMass** Halo mass [$M_{\odot}$];
- **wavenumber** Wavenumber corresponding to this halo mass [Mpc$^{-1}$];
- **powerSpectrum** Power spectrum at this wavenumber [Mpc$^3$];
- **variance** The variance, $S(M) = \sigma^2(M)$, at this halo mass;
- **barrier** The excursion set barrier, $B(S)$;
- **firstCrossingProbability** The probability of first crossing this barrier between $S$ and $S + dS$;
- **firstCrossingRate** The rate of first crossing of the barrier per unit time [Gyr$^{-1}$] for all pairs of halo mass;
- **haloMassFunction** The halo mass function [$M_{\odot}^{-1}$ Mpc$^{-3}$].

#### 3.4.2. Halo_Mass_Functions

The **Halo_Mass_Functions** code will generate an HDF5 output file which contains a variety of measures of the dark matter halo mass function tabulated as a function of mass and at a variety of redshifts. The code is built and run as follows:

```
make Halo_Mass_Functions.exe
Halo_Mass_Functions.exe <parameterFile> <outputFile>
```

where `parameterFile` is a file of parameters in **Galacticus**’s usual XML format and `outputFile` is the name of the file to which the halo mass function data will be written. The parameter file can specify any parameters needed for computing the mass function (they will be set to default values in cases where a parameter is not included). The redshifts at which to output halo mass functions are given by the [outputRedshifts] parameter. In addition to the usual **Galacticus** parameters three additional parameters control behavior:

- **[haloMassFunctionsMassMinimum]** The lowest mass halo (in units of $M_{\odot}$) at which to tabulate;
- **[haloMassFunctionsMassMaximum]** The highest mass halo (in units of $M_{\odot}$) at which to tabulate;
- **[haloMassFunctionsPointsPerDecade]** The number of points per decade of halo mass at which to tabulate.

The output file has the following structure:

```
|  
| +-> Outputs  
|   
|   |  
|   | +-> outputCharacteristicMass [dataset]  
|   |  
|   | +-> outputCriticalOverdensities [dataset]  
```
3.4. Additional Codes

The Parameters group contains attributes giving the values of all used parameters (just as in a GALACTICUS output file). The Outputs group contains datasets which give global properties at each requested output time as follows:

outputCharacteristicMass The characteristic mass scale (in units of $M_\odot$), $M_\ast$, at which $\sigma(M) = \delta_c(z)$;
outputCriticalOverdensities The critical overdensity for collapse of halos, $\delta_c$;
outputExpansionFactor The expansion factor;
outputGrowthFactors The linear growth factor;
3. Running Galacticus

outputRedshift The redshift;
outputTime The cosmic time (in units of Gyr);
outputVirialDensityContrast The virial density contrast of halos.

The haloMassFunctions group contains datasets which list the properties of halos as a function of mass at each requested output time as follows:

haloBias The large scale linear theory bias of the halo;
haloMass The mass of the halo (in \( M_\odot \));
haloMassFractionCumulative The mass fraction in halos above the current halo mass;
haloMassFunctionCumulative The cumulative number of halos per unit volume above the current halo mass (in units of Mpc\(^{-3} \));
haloMassFunctionLnM The halo mass function per logarithmic halo mass (in units of Mpc\(^{-3} \));
haloMassFunctionM The halo mass function per logarithmic halo mass (in units of Mpc\(^{-3} M_\odot^{-1} \));
haloNu The peak height of the halo, \( \nu = \delta_c/\sigma(M) \);
haloSigma The root-variance of the mass field smoothed in top-hat spheres;
haloVirialRadius The virial radius (in units of Mpc) of the current halo mass;
haloVirialTemperature The virial temperature (in units of Kelvin) of the current halo mass;
haloVirialVelocity The virial velocity (in units of km/s) of the current halo mass;

Dimensionful datasets have an `unitsInSI` attribute that gives their units in the SI system.

3.4.3. Power_Spectra

The Power_Spectra code will generate an HDF5 output file which contains a variety of measures of the matter power spectrum tabulated as a function of wavenumber. The code is built and run as follows:

```
make Power_Spectra.exe
Power_Spectra.exe <parameterFile> <outputFile>
```

where `parameterFile` is a file of parameters in GALACTICUS’s usual XML format and `outputFile` is the name of the file to which the power spectrum data will be written. The parameter file can specify any parameters needed for computing the power spectrum (they will be set to default values in cases where a parameter is not included). The output file has the following structure:

```
+-> powerSpectrum
  |  |
  |  +-> alpha [dataset]
  |  |
  |  +-> mass [dataset]
  |  |
  |  +-> powerSpectrum [dataset]
  |  |
  |  +-> sigma [dataset]
```
The Parameters group contains attributes giving the values of all used parameters (just as in a GALACTICUS output file). The powerSpectrum group contains datasets which give the power spectrum and related properties as follows:

**alpha** The logarithmic slope of $\sigma(M)$: $\alpha = \frac{d \ln \sigma}{d \ln M}$;

**mass** The mass scale, $M$, corresponding to the given wavenumber, $k$, defined such that $M = \frac{4\pi\Omega_{\text{M}}\rho_{\text{crit}}}{3k^3}$ (in units of $M_\odot$);

**powerSpectrum** The linear theory power spectrum at $z = 0$: $P(k)$ in units of $\text{Mpc}^3$;

**sigma** The dimensionless linear theory mass fluctuation at $z = 0$: $\sigma(M)$;

**wavenumber** The wavenumber in units of $\text{Mpc}^{-1}$.

Dimensionful datasets have an **unitsInSI** attribute that gives their units in the SI system.
4. Input Parameters

The following is an alphnumerically sorted list of all input parameters defined in GALACTICUS. Each parameter is listed by name, along with a description, default value (if one is specified in GALACTICUS), the file and program unit with which it is associated. Where relevant, references for parameters and the default values are given.

Name: tfamily H\_0
**Attached to:** tfamily module:Cosmology_Parameters
**File:** tfamily cosmology.parameters.F90
**Default value:** 69.7 (Hinshaw et al. 2012; CMB + H\_0 + BAO)
**Description:** The present day value of the Hubble parameter in units of km/s/Mpc.

Name: tfamily Omega\_DE
**Attached to:** tfamily module:Cosmology_Parameters
**File:** tfamily cosmology.parameters.F90
**Default value:** 0.7188 (Hinshaw et al. 2012; CMB + H\_0 + BAO)
**Description:** The density of dark energy in the Universe in units of the critical density.

Name: tfamily Omega\_Matter
**Attached to:** tfamily module:Cosmology_Parameters
**File:** tfamily cosmology.parameters.F90
**Default value:** 0.2812 (Hinshaw et al. 2012; CMB + H\_0 + BAO)
**Description:** The density of matter in the Universe in units of the critical density.

Name: tfamily Omega\_b
**Attached to:** tfamily module:Cosmology_Parameters
**File:** tfamily cosmology.parameters.F90
**Default value:** 0.04611 (Hinshaw et al. 2012; CMB + H\_0 + BAO)
**Description:** The density of baryons in the Universe in units of the critical density.

Name: tfamily T\_CMB
**Attached to:** tfamily module:Cosmology_Parameters
**File:** tfamily cosmology.parameters.F90
**Default value:** 2.72548 [Fixsen, 2009]
**Description:** The present day temperature of the cosmic microwave background (CMB) in units of Kelvin.

**Attached to:** tfamily module:Stellar_Population_Spectra_File
**File:** tfamily stellar_populations.spectra.file.F90
**Default value:** data/SSP_Spectra_imf$1.hdf5
**Description:** The name of the file of stellar populations to use for the named initial mass function (IMF).

Name: tfamily [regEx] stellarPopulationSpectraPostprocess[a-zA-Z0-9\_]+Methods
**Attached to:** tfamily module:Stellar_Population_Spectra_Postprocess
4. Input Parameters

File: tfamily stellar_populations.spectra.postprocess.F90
Default value: inoue2014 (for “Default” chain)
Description: The name of methods to be used for post-processing of stellar population spectra.

Name: tfamily accretionColdModeShockStabilityThreshold
Attached to: tfamily module:Accretion_Halos
File: tfamily accretion.halo.F90
Default value: 0.0126 [Birnboim and Dekel, 2003]
Description: The threshold value, \( \epsilon_{s,\text{crit}} \), for shock stability in the model of Birnboim and Dekel [2003].

Name: tfamily accretionColdModeShockStabilityTransitionWidth
Attached to: tfamily module:Accretion_Halos
File: tfamily accretion.halo.F90
Default value: 0.01 [Benson and Bower, 2010]
Description: The width of the transition from stability to instability for cold mode accretion [Benson and Bower, 2010].

Name: tfamily accretionDiskJetPowerEddington
Attached to: tfamily module:Accretion_Disks_Eddington
File: tfamily accretion_disks.Eddington_limited.F90
Default value: 0.1
Description: The jet power produced by an Eddington limited accretion disk in units of the Eddington luminosity.

Name: tfamily accretionDiskRadiativeEfficiencyEddington
Attached to: tfamily module:Accretion_Disks_Eddington
File: tfamily accretion_disks.Eddington_limited.F90
Default value: 0.1
Description: The radiative efficiency of an Eddington limited accretion disk.

Name: tfamily accretionDiskSpectraFileName
Attached to: tfamily module:Accretion_Disk_Spectra
File: tfamily accretion_disks.spectra.F90
Default value: none
Description: The name of a file from which to read tabulated spectra of accretion disks.

Name: tfamily accretionDiskSpectraMethod
Attached to: tfamily module:Accretion_Disk_Spectra
File: tfamily accretion_disks.spectra.F90
Default value: hopkins2007
Description: The method to be used for accretionDiskSpectra.

Name: tfamily accretionDiskSwitchedScaleAdafRadiativeEfficiency
Attached to: tfamily module:Accretion_Disks_Switched
File: tfamily accretion_disks.switched.F90
Default value: true
Description: Specifies whether the radiative efficiency of the ADAF component in a switched accretion disk scales with accretion rate.
Name: tfamily accretionDisksMethod
Attached to: tfamily module:Accretion_Disks
File: tfamily accretion_disks.F90
Default value: switched
Description: Selects which accretion disk method should be used.

Name: tfamily accretionHaloMethod
Attached to: tfamily module:Accretion_Halos
File: tfamily accretion.halo.F90
Default value: simple
Description: The method to be used for accretionHalo.

Name: tfamily accretionHalosSimpleAccreteNewGrowthOnly
Attached to: tfamily module:Accretion_Halos
File: tfamily accretion.halo.F90
Default value: false
Description: Specifies whether accretion from the intergalactic medium (IGM) is allowed only when a halo is growing past its previous greatest mass.

Name: tfamily accretionHalosSimpleNegativeAccretionAllowed
Attached to: tfamily module:Accretion_Halos
File: tfamily accretion.halo.F90
Default value: true
Description: Specifies whether negative accretion (mass loss) is allowed in the simple halo accretion model.

Name: tfamily accretionHistoryWechslerFormationRedshift
Attached to: tfamily module:Dark_Matter_Halo_Mass_Accretion_Histories_Wechsler2002
Default value: 0.4
Description: The formation redshift to use in Wechsler et al. [2002] halo mass accretion histories.

Name: tfamily accretionHistoryWechslerFormationRedshiftCompute
Attached to: tfamily module:Dark_Matter_Halo_Mass_Accretion_Histories_Wechsler2002
Default value: true
Description: Compute formation redshift automatically for Wechsler et al. [2002] halo mass accretion histories?

Name: tfamily accretionRateThinDiskMaximum
Attached to: tfamily module:Accretion_Disks_Switched
File: tfamily accretion_disks.switched.F90
Default value: 0.3
Description: The accretion rate (in Eddington units) above which a switched accretion disk becomes an ADAF.

Name: tfamily accretionRateThinDiskMinimum
Attached to: tfamily module:Accretion_Disks_Switched
File: tfamily accretion_disks.switched.F90
4. Input Parameters

**Default value:** 0.01  
**Description:** The accretion rate (in Eddington units) below which a switched accretion disk becomes an ADAF.

**Name:** tfamily accretionRateTransitionWidth  
**Attached to:** tfamily module:Accretion_Disks_Switched  
**File:** tfamily accretion_disks.switched.F90  
**Default value:** 0.1  
**Description:** The width (in \( \ln[M/\dot{M}_{\text{Eddington}}] \)) over which transitions between accretion disk states occur.

**Name:** tfamily adafAdiabaticIndex  
**Attached to:** tfamily module:Accretion_Disks_ADAF  
**File:** tfamily accretion_disks.ADAF.F90  
**Default value:** 1.444 (for exponential form of field-enhancing shear) or 1.333 (for linear form)  
**Description:** Specifies the effective adiabatic index of gas in an ADAF.

**Name:** tfamily adafEnergyOption  
**Attached to:** tfamily module:Accretion_Disks_ADAF  
**File:** tfamily accretion_disks.ADAF.F90  
**Default value:** pureADAF  
**Description:** Specifies the specific energy of material at the inner edge of an ADAF. pureADAF makes the specific energy equal to 1 (i.e. all energy is advected with the flow); ISCO makes the specific energy equal to that for the innermost stable circular orbit.

**Name:** tfamily adafFieldEnhanceType  
**Attached to:** tfamily module:Accretion_Disks_ADAF  
**File:** tfamily accretion_disks.ADAF.F90  
**Default value:** exponential  
**Description:** Controls how the field enhancing shear is determined. exponential will cause the form \( g = \exp(\omega t) \) [Benson and Babul, 2009] to be used, while linear will cause \( g = 1 + \omega t \) to be used instead. The functional form of \( \alpha(j) \) (if used) will be adjusted to achieve a sensible spin-up function in each case.

**Name:** tfamily adafJetEfficiencyMaximum  
**Attached to:** tfamily module:Accretion_Disks_ADAF  
**File:** tfamily accretion_disks.ADAF.F90  
**Default value:** 2  
**Description:** The maximum efficiency allowed for ADAF-driven jets (in units of the accretion power).

**Name:** tfamily adafRadiativeEfficiency  
**Attached to:** tfamily module:Accretion_Disks_ADAF  
**File:** tfamily accretion_disks.ADAF.F90  
**Default value:** 0.01  
**Description:** Specifies the radiative efficiency of an ADAF (i.e. the fraction of \( \dot{M}c^2 \) that is emitted in radiation).

**Name:** tfamily adafRadiativeEfficiencyType  
**Attached to:** tfamily module:Accretion_Disks_ADAF  
**File:** tfamily accretion_disks.ADAF.F90
**Default value:** thinDisk  
**Description:** Specifies the specific energy of material at the inner edge of an ADAF. pureADAF makes the specific energy equal to 1 (i.e. all energy is advected with the flow); ISCO makes the specific energy equal to that for the innermost stable circular orbit.

**Name:** tfamily adafViscosityFixedAlpha  
**Attached to:** tfamily module:Accretion_Disks_ADAF  
**File:** tfamily accretion_disks.ADAF.F90  
**Default value:** 0.1  
**Description:** The value for the viscosity parameter $\alpha$ in an ADAF to be used if [adafViscosityOption]=fixed.

**Name:** tfamily adafViscosityOption  
**Attached to:** tfamily module:Accretion_Disks_ADAF  
**File:** tfamily accretion_disks.ADAF.F90  
**Default value:** fit  
**Description:** Controls how the viscosity parameter $\alpha$ in an ADAF is determined. fit will cause $\alpha$ to be computed using the fitting function of Benson and Babul [2009]; fixed will cause $\alpha = [adafViscosityFixedAlpha]$ to be used.

**Name:** tfamily adiabaticContractionGnedinA  
**Attached to:** tfamily module:Galactic_Structure_Initial_Radii_Adiabatic  
**File:** tfamily galactic_structure.radius_solver.initial_radii.adiabatic.F90  
**Default value:** 0.8 (Gustafsson et al. 2006; from their Fig. 9, strong feedback case)  
**Description:** The parameter $A$ appearing in the Gnedin et al. [2004] adiabatic contraction algorithm.

**Name:** tfamily adiabaticContractionGnedinOmega  
**Attached to:** tfamily module:Galactic_Structure_Initial_Radii_Adiabatic  
**File:** tfamily galactic_structure.radius_solver.initial_radii.adiabatic.F90  
**Default value:** 0.77 (Gustafsson et al. 2006; from their Fig. 9, strong feedback case)  
**Description:** The parameter $\omega$ appearing in the Gnedin et al. [2004] adiabatic contraction algorithm.

**Name:** tfamily adiabaticContractionIncludeBaryonGravity  
**Attached to:** tfamily module:Galactic_Structure_Radii_Adiabatic  
**File:** tfamily galactic_structure.radius_solver.adiabatic.F90  
**Default value:** true  
**Description:** Specifies whether or not gravity from baryons is included when solving for sizes of galactic components in adiabatically contracted dark matter halos.

**Name:** tfamily adiabaticContractionSolutionTolerance  
**Attached to:** tfamily module:Galactic_Structure_Radii_Adiabatic  
**File:** tfamily galactic_structure.radius_solver.adiabatic.F90  
**Default value:** $10^{-2}$  
**Description:** Maximum allowed mean fractional error in the radii of all components when seeking equilibrium solutions for galactic structure.

**Name:** tfamily adiabaticContractionUseFormationHalo  
**Attached to:** tfamily module:Galactic_Structure_Radii_Adiabatic  
**File:** tfamily galactic_structure.radius_solver.adiabatic.F90  
**Default value:** false

41
4. Input Parameters

**Description:** Specifies whether or not the “formation halo” should be used when solving for the radii of galaxies.

**Name:** tfamily alfalfaHiMassFunctionZ0.00ConversionError  
**Attached to:** tfamily module:Galacticus_Output_Analyses_Mass_Functions  
**File:** tfamily galacticus.output.analysis.mass_functions.F90  
**Default value:** 0.4  
**Description:** The random error (in dex) to be added to galaxy HI masses when constructing the ALFALFA HI mass function. This error accounts for scatter in the H$_2$/HI mass ratio at fixed total gas mass.

**Name:** tfamily alfalfaHiMassFunctionZ0.00MolecularFractionKappa  
**Attached to:** tfamily module:Galacticus_Output_Analyses_Mass_Functions  
**File:** tfamily galacticus.output.analysis.mass_functions.F90  
**Default value:** 1.3  
**Description:** The parameter, $\kappa$, appearing in the model for the H$_2$/HI mass ratio used when constructing the ALFALFA HI mass function. Specifically, $\log_{10} R_{mol} = \mu + \kappa \log_{10} (M_{gas}/10^9 M_\odot)$.

**Name:** tfamily alfalfaHiMassFunctionZ0.00MolecularFractionMu  
**Attached to:** tfamily module:Galacticus_Output_Analyses_Mass_Functions  
**File:** tfamily galacticus.output.analysis.mass_functions.F90  
**Default value:** 0.9  
**Description:** The parameter, $\mu$, appearing in the model for the H$_2$/HI mass ratio used when constructing the ALFALFA HI mass function. Specifically, $\log_{10} R_{mol} = \mu + \kappa \log_{10} (M_{gas}/10^9 M_\odot)$.

**Name:** tfamily allTreesExistAtFinalTime  
**Attached to:** tfamily module:Merger_Trees_Evolve  
**File:** tfamily merger_trees.evolve.F90  
**Default value:** true  
**Description:** Specifies whether or not all merger trees are expected to exist at the final requested output time. If set to false, then trees which finish before a given output time will be ignored.

**Name:** tfamily analysisMassFunctionCovarianceModel  
**Attached to:** tfamily module:Galacticus_Output_Analyses_Mass_Functions  
**File:** tfamily galacticus.output.analysis.mass_functions.F90  
**Default value:** binomial  
**Description:** The model to use when constructing the mass function covariance matrix for main branch galaxies.

**Name:** tfamily analysisMassFunctionsCorrelationTruncateLevel  
**Attached to:** tfamily module:Galacticus_Output_Analyses_Mass_Functions  
**File:** tfamily galacticus.output.analysis.mass_functions.F90  
**Default value:** 0.0  
**Description:** The correlation below which off-diagonal elements of the covariance matrix are truncated to zero.

**Name:** tfamily analysisMassFunctionsHaloMassBinsPerDecade  
**Attached to:** tfamily module:Galacticus_Output_Analyses_Mass_Functions  
**File:** tfamily galacticus.output.analysis.mass_functions.F90  
**Default value:** 10
**Description:** The number of bins per decade of halo mass to use when constructing the mass function covariance matrix for main branch galaxies.

**Name:** tfamily analysisMassFunctionsHaloMassBinsPerDecade
**Attached to:** tfamily module:Galacticus_Output_Analyses_Mass_Functions
**File:** tfamily galacticus.output.analysis.mass_functions.F90
**Default value:** $10^{16} M_\odot$
**Description:** The maximum halo mass to consider when constructing the mass function covariance matrix for main branch galaxies.

**Name:** tfamily analysisMassFunctionsHaloMassMinimum
**Attached to:** tfamily module:Galacticus_Output_Analyses_Mass_Functions
**File:** tfamily galacticus.output.analysis.mass_functions.F90
**Default value:** $10^8 M_\odot$
**Description:** The minimum halo mass to consider when constructing the mass function covariance matrix for main branch galaxies.

**Name:** tfamily analysisSizeFunctionCovarianceModel
**Attached to:** tfamily module:Galacticus_Output_Analyses_Mass_Dpndnt_Sz_Dstrbtins
**File:** tfamily galacticus.output.analysis.mass_dependent_size_distribution.F90
**Default value:** binomial
**Description:** The model to use when constructing the mass function covariance matrix for main branch galaxies.

**Name:** tfamily analysisSizeFunctionsHaloMassBinsPerDecade
**Attached to:** tfamily module:Galacticus_Output_Analyses_Mass_Dpndnt_Sz_Dstrbtins
**File:** tfamily galacticus.output.analysis.mass_dependent_size_distribution.F90
**Default value:** 10
**Description:** The number of bins per decade of halo mass to use when constructing the mass function covariance matrix for main branch galaxies.

**Name:** tfamily analysisSizeFunctionsHaloMassMaximum
**Attached to:** tfamily module:Galacticus_Output_Analyses_Mass_Dpndnt_Sz_Dstrbtins
**File:** tfamily galacticus.output.analysis.mass_dependent_size_distribution.F90
**Default value:** $10^{16} M_\odot$
**Description:** The maximum halo mass to consider when constructing the mass function covariance matrix for main branch galaxies.

**Name:** tfamily analysisSizeFunctionsHaloMassMinimum
**Attached to:** tfamily module:Galacticus_Output_Analyses_Mass_Dpndnt_Sz_Dstrbtins
**File:** tfamily galacticus.output.analysis.mass_dependent_size_distribution.F90
**Default value:** $10^8 M_\odot$
**Description:** The minimum halo mass to consider when constructing the mass function covariance matrix for main branch galaxies.

**Name:** tfamily atomicCollisionalIonizationMethod
**Attached to:** tfamily module:Atomic_Rates_Ionization_Collisional
**File:** tfamily atomic.rates.ionization.collisional.F90
**Default value:** Verner

43
4. Input Parameters

**Description:** The name of the method to be used for computing atomic collisional ionization rates.

**Name:** tfamily atomicIonizationPotentialMethod  
**Attached to:** tfamily module: Atomic_Ionization_Potentials  
**File:** tfamily atomic.ionization_potentials.F90  
**Default value:** Verner  
**Description:** The name of the method to be used for computing atomic ionization potentials.

**Name:** tfamily atomicPhotoIonizationMethod  
**Attached to:** tfamily module; Atomic_Cross_Sections_Ionization_Photo  
**File:** tfamily atomic.cross_sections.ionization/photo.F90  
**Default value:** Verner  
**Description:** The name of the method to be used for computing atomic photo ionization rates.

**Name:** tfamily atomicRadiativeRecombinationMethod  
**Attached to:** tfamily module: Atomic_Rates_Recombination_Radiative  
**File:** tfamily atomic.rates.recombination.radiative.F90  
**Default value:** Verner  
**Description:** The name of the method to be used for computing atomic radiative recombination rates.

**Name:** tfamily backgroundRadiationCompute  
**Attached to:** tfamily module: Radiation_Intergalactic_Background_Internal  
**File:** tfamily radiation.intergalactic_background.internal.F90  
**Default value:** false  
**Description:** Specifies whether or not cosmic background radiation should be computed.

**Name:** tfamily backgroundRadiationRedshiftMaximum  
**Attached to:** tfamily module: Radiation_Intergalactic_Background_Internal  
**File:** tfamily radiation.intergalactic_background.internal.F90  
**Default value:** 30  
**Description:** The maximum redshift to use in calculations of the cosmic background radiation.

**Name:** tfamily backgroundRadiationRedshiftMinimum  
**Attached to:** tfamily module: Radiation_Intergalactic_Background_Internal  
**File:** tfamily radiation.intergalactic_background.internal.F90  
**Default value:** 0  
**Description:** The minimum redshift to use in calculations of the cosmic background radiation.

**Name:** tfamily backgroundRadiationTimeCountPerDecade  
**Attached to:** tfamily module: Radiation_Intergalactic_Background_Internal  
**File:** tfamily radiation.intergalactic_background.internal.F90  
**Default value:** 10  
**Description:** The number of bins per decade of time to use for calculations of the cosmic background radiation.

**Name:** tfamily backgroundRadiationWavelengthCountPerDecade  
**Attached to:** tfamily module: Radiation_Intergalactic_Background_Internal  
**File:** tfamily radiation.intergalactic_background.internal.F90  
**Default value:** 10
**Description**: The number of bins per decade of wavelength to use for calculations of the cosmic background radiation.

**Name**: tfamily backgroundRadiationWavelengthMaximum  
**Attached to**: tfamily module:Radiation_Intergalactic_Background_Internal  
**File**: tfamily radiation.intergalactic_background.internal.F90  
**Default value**: 100000 Å  
**Description**: The maximum wavelength (in units of Å) to use in calculations of the cosmic background radiation.

**Name**: tfamily backgroundRadiationWavelengthMinimum  
**Attached to**: tfamily module:Radiation_Intergalactic_Background_Internal  
**File**: tfamily radiation.intergalactic_background.internal.F90  
**Default value**: 100 Å  
**Description**: The minimum wavelength (in units of Å) to use in calculations of the cosmic background radiation.

**Name**: tfamily barInstabilityMethod  
**Attached to**: tfamily module:Galactic_Dynamics_Bar_Instabilities  
**File**: tfamily galactic_dynamics.bar_instability.F90  
**Default value**: ELN  
**Description**: The name of the method to be used for bar instability calculations.

**Name**: tfamily blackHoleAccretesFromHotHalo  
**Attached to**: tfamily module:Node_Component_Black_Hole_Simple  
**File**: tfamily objects.nodes.components.black_hole.simple.F90  
**Default value**: false  
**Description**: Controls whether the black hole additionally grows via accretion from the hot halo. If it does, this accretion rate is used to determine AGN feedback power.

**Name**: tfamily blackHoleBinariesComputeVelocityDispersion  
**Attached to**: tfamily module:Black_Hole_Binary_Separations_Standard  
**File**: tfamily black_holes.binaries.separation_growth_rate.standard.F90  
**Default value**: false  
**Description**: Specifies whether or not the velocity dispersion of dark matter and stars should be computed using Jeans equation in black hole binary hardening calculations. If **false**, then the velocity dispersions are assumed to equal the characteristic velocity of dark matter and spheroid.

**Name**: tfamily blackHoleBinaryInitialRadiiMethod  
**Attached to**: tfamily module:Black_Hole_Binary_Initial_Radii  
**File**: tfamily black_holes.binaries.initial_radius.F90  
**Default value**: spheroidRadiusFraction  
**Description**: The name of the method to be used for computing the initial separation of black hole binaries.

**Name**: tfamily blackHoleBinaryMergersMethod  
**Attached to**: tfamily module:Black_Hole_Binary_Mergers  
**File**: tfamily black_holes.binary_mergers.F90  
**Default value**: Rezzolla2008
4. Input Parameters

**Description:** The name of the method to be used for computing the effects of black hole binary mergers.

**Name:** tfamily blackHoleBinaryRecoilVelocityMethod  
**Attached to:** tfamily module:Black_Hole_Binary_Recoil_Velocities  
**File:** tfamily black_holes.binaries.recoil_velocity.F90  
**Default value:** null  
**Description:** The name of the method to be used for computing the recoil velocity of black hole binaries.

**Name:** tfamily blackHoleBinarySeparationGrowthRateMethod  
**Attached to:** tfamily module:Black_Hole_Binary_Separations  
**File:** tfamily black_holes.binaries.separation_growth_rate.F90  
**Default value:** null  
**Description:** The name of the method to be used for computing the separation growth rate of black hole binaries.

**Name:** tfamily blackHoleHeatingEfficiency  
**Attached to:** tfamily module:Node_Component_Black_Hole_Simple  
**File:** tfamily objects.nodes.components.black_hole.simple.F90  
**Default value:** $10^{-3}$  
**Description:** The efficiency with which accretion onto a black hole heats the hot halo.

**Name:** tfamily blackHoleHeatsHotHalo  
**Attached to:** tfamily module:Node_Component_Black_Hole_Simple  
**File:** tfamily objects.nodes.components.black_hole.simple.F90  
**Default value:** true  
**Description:** Specifies whether or not the black hole should heat the hot halo.

**Name:** tfamily blackHoleInitialRadiusSpheroidRadiusRatio  
**Attached to:** tfamily module:Black_Hole_Binary_Initial_Radii_Spheroid_Size  
**File:** tfamily black_holes.binaries.initial_radius.spheroid_size_fraction.F90  
**Default value:** 0  
**Description:** The fraction of the spheroid radius at which merging black holes will be initially placed.

**Name:** tfamily blackHoleJetEfficiency  
**Attached to:** tfamily module:Node_Component_Black_Hole_Simple  
**File:** tfamily objects.nodes.components.black_hole.simple.F90  
**Default value:** $10^{-3}$  
**Description:** The efficiency with which accretion power onto a black hole is converted into jets.

**Name:** tfamily blackHoleOutputAccretion  
**Attached to:** tfamily module:Node_Component_Black_Hole_Simple  
**File:** tfamily objects.nodes.components.black_hole.simple.F90  
**Default value:** false  
**Description:** Determines whether or not accretion rates and jet powers will be output.

**Name:** tfamily blackHoleOutputData  
**Attached to:** tfamily module:Node_Component_Black_Hole_Standard  
**File:** tfamily objects.nodes.components.black_hole.standard.F90  
**Default value:** false
**Description:** Determines whether or not properties for all black holes (rather than just the central black hole) will be output.

**Name:** tfamily blackHoleOutputMergers  
**Attached to:** tfamily module:Node_Component_Black_Hole_Standard  
**File:** tfamily objects.nodes.components.black_hole.standard.F90  
**Default value:** false  
**Description:** Determines whether or not properties of black hole mergers will be output.

**Name:** tfamily blackHoleRadioModeFeedbackEfficiency  
**Attached to:** tfamily module:Node_Component_Black_Hole_Standard  
**File:** tfamily objects.nodes.components.black_hole.standard.F90  
**Default value:** 1  
**Description:** Efficiency with which radio-mode feedback is coupled to the hot halo.

**Name:** tfamily blackHoleSeedMass  
**Attached to:** tfamily module:Galacticus_Nodes  
**File:** tfamily objects.nodes.F90  
**Default value:** 100  
**Description:** The mass of the seed black hole placed at the center of each newly formed galaxy.

**Name:** tfamily blackHoleToSpheroidStellarGrowthRatio  
**Attached to:** tfamily module:Node_Component_Black_Hole_Simple  
**File:** tfamily objects.nodes.components.black_hole.simple.F90  
**Default value:** 1.0d-3  
**Description:** The ratio of the rates of black hole growth and spheroid stellar mass growth.

**Name:** tfamily blackHoleWindEfficiency  
**Attached to:** tfamily module:Node_Component_Black_Hole_Simple  
**File:** tfamily objects.nodes.components.black_hole.simple.F90  
**Default value:** 2.2157 × 10⁻³  
**Description:** The efficiency of the black hole accretion-driven wind.

**Name:** tfamily blackHoleWindEfficiencyScalesWithRadiativeEfficiency  
**Attached to:** tfamily module:Node_Component_Black_Hole_Standard  
**File:** tfamily objects.nodes.components.black_hole.standard.F90  
**Default value:** false  
**Description:** Specifies whether the black hole wind efficiency should scale with the radiative efficiency of the accretion disk.

**Name:** tfamily bondiHoyleAccretionEnhancementHotHalo  
**Attached to:** tfamily module:Node_Component_Black_Hole_Standard  
**File:** tfamily objects.nodes.components.black_hole.standard.F90  
**Default value:** 6.0  
**Description:** The factor by which the Bondi-Hoyle accretion rate of hot halo gas onto black holes in enhanced.

**Name:** tfamily bondiHoyleAccretionEnhancementSpheroid  
**Attached to:** tfamily module:Node_Component_Black_Hole_Standard
4. Input Parameters

**Name:** tfamily bondiHoyleAccretionHotModeOnly
**Attached to:** tfamily module:Node_Component_Black_Hole_Standard
**File:** tfamily objects.nodes.components.black_hole.standard.F90
**Default value:** true
**Description:** Determines whether accretion from the hot halo should only occur if the halo is in the hot accretion mode.

**Name:** tfamily bondiHoyleAccretionTemperatureSpheroid
**Attached to:** tfamily module:Node_Component_Black_Hole_Standard
**File:** tfamily objects.nodes.components.black_hole.standard.F90
**Default value:** $10^2$
**Description:** The assumed temperature (in Kelvin) of gas in the spheroid when computing Bondi-Hoyle accretion rates onto black holes.

**Name:** tfamily boxSize
**Attached to:** tfamily module:Merger_Trees_Simple
**File:** tfamily merger_trees.file_maker.simple.F90
**Default value:** none
**Description:** The box size of the simulation from which merger trees were extracted.

**Name:** tfamily burstCriticalGasFraction
**Attached to:** tfamily module:Satellite_Merging_Mass_Movements_Baugh2005
**File:** tfamily satellites.merging.mass_movements.Baugh2005.F90
**Default value:** 0.75
**Description:** The host gas fraction above which mergers are considered to trigger a burst in the Baugh et al. [2005] merger mass movements method.

**Name:** tfamily burstMassRatio
**Attached to:** tfamily module:Satellite_Merging_Mass_Movements_Baugh2005
**File:** tfamily satellites.merging.mass_movements.Baugh2005.F90
**Default value:** 0.05
**Description:** The mass ratio above which mergers are considered to trigger a burst in the Baugh et al. [2005] merger mass movements method.

**Name:** tfamily chemicalReactionRateMethods
**Attached to:** tfamily module:Chemical_Reaction_Rates
**File:** tfamily chemical.reaction_rates.F90
**Default value:** hydrogenNetwork
**Description:** The names of the methods to be used for computing chemical reaction rates.

**Name:** tfamily chemicalStateFile
**Attached to:** tfamily module:Chemical_States_CIE_File
**File:** tfamily chemical.state.CIE_file.F90
**Default value:** none
**Description:** The name of the file containing a tabulation of the collisional chemical equilibrium chemical state.

**Name:** tfamily chemicalStateMethod  
**Attached to:** tfamily module:Chemical_States  
**File:** tfamily chemical.state.F90  
**Default value:** atomicCIECloudy  
**Description:** The name of the method to be used for computing the chemical state.

**Name:** tfamily chemicalsToTrack  
**Attached to:** tfamily module:Chemical_Abundances_Structure  
**File:** tfamily objects.chemical_abundances.F90  
**Default value:** none  
**Description:** The names of the chemicals to be tracked.

**Name:** tfamily coldModeInfallRateDynamicalTime  
**Attached to:** tfamily module:Cooling_Cold_Mode_Infall_Rates_Dynamical_Time  
**File:** tfamily cooling.cold_mode.infall_rate.dynamical_time.F90  
**Default value:** 2.0  
**Description:** The timescale (in units of the halo dynamical time) for infall of the cold mode component.

**Name:** tfamily coldModeInfallRateMethod  
**Attached to:** tfamily module:Cooling_Cold_Mode_Infall_Rates  
**File:** tfamily cooling.cold_mode.infall_rate.F90  
**Default value:** dynamicalTime  
**Description:** The name of the method to be used when computing the infall rate from the cold mode.

**Name:** tfamily coldModeIsothermalCoreRadiusOverVirialRadius  
**Attached to:** tfamily module:Hot_Halo_Cold_Mode_Density_CIsolatedCoreR_Virial_Fraction  
**File:** tfamily hot_halo.cold_mode.density_profile.cored_isothermal.core_radius.virial_radius.F90  
**Default value:** 0.3  
**Description:** The core radius in the “cored isothermal” cold mode hot halo density profile in units of the virial radius.

**Name:** tfamily coldModeMassDistribution  
**Attached to:** tfamily module:Node_Component_Hot_Halo_Cold_Mode  
**File:** tfamily objects.nodes.components.hot_halo.cold_mode.F90  
**Default value:** betaProfile  
**Description:** The type of mass distribution to use for the cold mode component.

**Name:** tfamily collisionalExcitationMethod  
**Attached to:** tfamily module:Atomic_Rates_Excitation_Collisional  
**File:** tfamily atomic.rates.excitation.collisional.F90  
**Default value:** ScholzWalters91  
**Description:** The name of the method to be used for computing collisional excitation rates.

**Name:** tfamily conditionalMassFunctionBehrooziAlphaSatellite  
**Attached to:** tfamily module:Conditional_Mass_Functions_Behroozi2010  
**File:** tfamily halo_model.conditional_mass_function.Behroozi2010.F90
4. Input Parameters

**Default value:** 1.0 (Leauthaud et al. 2011; $z_1$ sample using their SIG_MOD1 method)
**Description:** The parameter $\alpha_s$ from the fitting functions of Behroozi et al. [2010].

**Name:** tfamily conditionalMassFunctionBehrooziBCut
**Attached to:** tfamily module:Conditional_Mass_Functions_Behroozi2010
**File:** tfamily halo_model.conditional_mass_function.Behroozi2010.F90
**Default value:** 1.47 (Leauthaud et al. 2011; $z_1$ sample using their SIG_MOD1 method)
**Description:** The parameter $B_{cut}$ from the fitting functions of Behroozi et al. [2010].

**Name:** tfamily conditionalMassFunctionBehrooziBSatellite
**Attached to:** tfamily module:Conditional_Mass_Functions_Behroozi2010
**File:** tfamily halo_model.conditional_mass_function.Behroozi2010.F90
**Default value:** 10.62 (Leauthaud et al. 2011; $z_1$ sample using their SIG_MOD1 method)
**Description:** The parameter $B_{sat}$ from the fitting functions of Behroozi et al. [2010].

**Name:** tfamily conditionalMassFunctionBehrooziBeta
**Attached to:** tfamily module:Conditional_Mass_Functions_Behroozi2010
**File:** tfamily halo_model.conditional_mass_function.Behroozi2010.F90
**Default value:** 0.457 (Leauthaud et al. 2011; $z_1$ sample using their SIG_MOD1 method)
**Description:** The parameter $\beta$ from the fitting functions of Behroozi et al. [2010].

**Name:** tfamily conditionalMassFunctionBehrooziBetaCut
**Attached to:** tfamily module:Conditional_Mass_Functions_Behroozi2010
**File:** tfamily halo_model.conditional_mass_function.Behroozi2010.F90
**Default value:** $-0.13$ (Leauthaud et al. 2011; $z_1$ sample using their SIG_MOD1 method)
**Description:** The parameter $\beta_{cut}$ from the fitting functions of Behroozi et al. [2010].

**Name:** tfamily conditionalMassFunctionBehrooziBetaSatellite
**Attached to:** tfamily module:Conditional_Mass_Functions_Behroozi2010
**File:** tfamily halo_model.conditional_mass_function.Behroozi2010.F90
**Default value:** 0.859 (Leauthaud et al. 2011; $z_1$ sample using their SIG_MOD1 method)
**Description:** The parameter $\beta_{sat}$ from the fitting functions of Behroozi et al. [2010].

**Name:** tfamily conditionalMassFunctionBehrooziDelta
**Attached to:** tfamily module:Conditional_Mass_Functions_Behroozi2010
**File:** tfamily halo_model.conditional_mass_function.Behroozi2010.F90
**Default value:** 0.5666 (Leauthaud et al. 2011; $z_1$ sample using their SIG_MOD1 method)
**Description:** The parameter $\delta$ from the fitting functions of Behroozi et al. [2010].

**Name:** tfamily conditionalMassFunctionBehrooziGamma
**Attached to:** tfamily module:Conditional_Mass_Functions_Behroozi2010
**File:** tfamily halo_model.conditional_mass_function.Behroozi2010.F90
**Default value:** 1.53 (Leauthaud et al. 2011; $z_1$ sample using their SIG_MOD1 method)
**Description:** The parameter $\gamma$ from the fitting functions of Behroozi et al. [2010].

**Name:** tfamily conditionalMassFunctionBehrooziLog10M1
**Attached to:** tfamily module:Conditional_Mass_Functions_Behroozi2010
**File:** tfamily halo_model.conditional_mass_function.Behroozi2010.F90
**Default value:** 12.520 (Leauthaud et al. 2011; $z_1$ sample using their SIG_MOD1 method)
Description: The parameter $\log_{10} M_1$ from the fitting functions of Behroozi et al. [2010].

Name: tfamily conditionalMassFunctionBehrooziLog10Mstar0
Attached to: tfamily module:Conditional_Mass_Functions_Behroozi2010
Default value: 10.916 (Leauthaud et al. 2011; $z_1$ sample using their SIG_MOD1 method)
Description: The parameter $\log_{10} M_{*,0}$ from the fitting functions of Behroozi et al. [2010].

Name: tfamily conditionalMassFunctionBehrooziSigmaLogMstar
Attached to: tfamily module:Conditional_Mass_Functions_Behroozi2010
Default value: 0.206 (Leauthaud et al. 2011; $z_1$ sample using their SIG_MOD1 method)
Description: The parameter $\sigma_{\log M_*}$ from the fitting functions of Behroozi et al. [2010].

Name: tfamily conditionalMassFunctionHaloMass
Attached to: tfamily program:Conditional_Mass_Function
File: tfamily Conditional_Mass_Function.F90
Default value: all
Description: The halo mass for which to compute the conditional mass function. A value of “all” will cause the conditional mass function to be integrated over the halo mass function, giving the mass function.

Name: tfamily conditionalMassFunctionHaloMassMaximum
Attached to: tfamily program:Conditional_Mass_Function
File: tfamily Conditional_Mass_Function.F90
Default value: $10^{16} M_\odot$
Description: The maximum halo mass to use when integrating over the halo mass function.

Name: tfamily conditionalMassFunctionHaloMassMinimum
Attached to: tfamily program:Conditional_Mass_Function
File: tfamily Conditional_Mass_Function.F90
Default value: $10^6 M_\odot$
Description: The minimum halo mass to use when integrating over the halo mass function.

Name: tfamily conditionalMassFunctionMassCount
Attached to: tfamily program:Conditional_Mass_Function
File: tfamily Conditional_Mass_Function.F90
Default value: 21
Description: The number of bins for which to compute the conditional mass function.

Name: tfamily conditionalMassFunctionMassMaximum
Attached to: tfamily program:Conditional_Mass_Function
File: tfamily Conditional_Mass_Function.F90
Default value: $10^{12} M_\odot$
Description: The maximum mass for which to compute the conditional mass function.

Name: tfamily conditionalMassFunctionMassMinimum
Attached to: tfamily program:Conditional_Mass_Function
File: tfamily Conditional_Mass_Function.F90
Default value: $10^8 M_\odot$
4. Input Parameters

**Description:** The minimum mass for which to compute the conditional mass function.

**Name:** tfamily conditionalMassFunctionMethod  
**Attached to:** tfamily module:Conditional_Mass_Functions  
**File:** tfamily halo_model.conditional_mass_function.F90  
**Default value:** Behroozi2010  
**Description:** The name of the method to be used for empirical models of the conditional mass function.

**Name:** tfamily conditionalMassFunctionOutputFileName  
**Attached to:** tfamily program:Conditional_Mass_Function  
**File:** tfamily Conditional_Mass_Function.F90  
**Default value:** none  
**Description:** The name of the file to which the computed conditional mass function should be output.

**Name:** tfamily conditionalMassFunctionRedshiftMaximum  
**Attached to:** tfamily program:Conditional_Mass_Function  
**File:** tfamily Conditional_Mass_Function.F90  
**Default value:** 0  
**Description:** The maximum redshift for which to compute the conditional mass function.

**Name:** tfamily conditionalMassFunctionRedshiftMinimum  
**Attached to:** tfamily program:Conditional_Mass_Function  
**File:** tfamily Conditional_Mass_Function.F90  
**Default value:** 0  
**Description:** The minimum redshift for which to compute the conditional mass function.

**Name:** tfamily conditionalMassFunctionUseSurveyLimits  
**Attached to:** tfamily program:Conditional_Mass_Function  
**File:** tfamily Conditional_Mass_Function.F90  
**Default value:** false  
**Description:** Specifies whether the limiting redshifts for integrating over the halo mass function should be limited by those of a galaxy survey.

**Name:** tfamily coolingAngularMomentumUseInteriorMean  
**Attached to:** tfamily module:Cooling_Specific_Angular_Momenta_Constant_Rotation  
**File:** tfamily cooling.specific-angular_momentum.constant_rotation.F90  
**Default value:** false  
**Description:** Specifies whether to use the specific angular momentum at the cooling radius, or the mean specific angular momentum interior to that radius.

**Name:** tfamily coolingCutOffFormationNode  
**Attached to:** tfamily module:Cooling_Rates_Modifier_Cut_Off  
**File:** tfamily cooling.cooling_rate.modifier.cut_off.F90  
**Default value:** false  
**Description:** Specifies whether to use the virial velocity of the formation node or current node in the cooling rate “cut-off” modifier.

**Name:** tfamily coolingCutOffRedshift  
**Attached to:** tfamily module:Cooling_Rates_Modifier_Cut_Off
File: tfamily cooling.cooling_rate.modifier.cut_off.F90
Default value: 0.0
Description: The redshift below which cooling is suppressed in the “cut-off” cooling rate modifier method.

Name: tfamily coolingCutOffVelocity
Attached to: tfamily module:Cooling_Rates_Modifier_Cut_Off
File: tfamily cooling.cooling_rate.modifier.cut_off.F90
Default value: 0.0
Description: The velocity below which cooling is suppressed in the “cut-off” cooling rate modifier method.

Name: tfamily coolingCutOffWhen
Attached to: tfamily module:Cooling_Rates_Modifier_Cut_Off
File: tfamily cooling.cooling_rate.modifier.cut_off.F90
Default value: after
Description: Specifies whether cooling is cut off before or after [coolingCutOffRedshift].

Name: tfamily coolingFunctionFile
Attached to: tfamily module:Cooling_Functions_CIE_File
File: tfamily cooling.cooling_function.CIE_file.F90
Default value: none
Description: The name of the file containing a tabulation of the collisional ionization equilibrium cooling function.

Name: tfamily coolingFunctionMethods
Attached to: tfamily module:Cooling_Functions
File: tfamily cooling.cooling_function.F90
Default value: atomicCIECloudy
Description: The names of the methods to be used for computing the cooling function.

Name: tfamily coolingMeanAngularMomentumFrom
Attached to: tfamily module:Cooling_Specific_Angular_Momenta_Constant_Rotation
File: tfamily cooling.specfic_angular_momentum.constant_rotation.F90
Default value: hotGas
Description: The component (“hotGas” or “darkMatter”) from which the mean specific angular momentum should be computed for calculations of cooling gas specific angular momentum.

Name: tfamily coolingRadiusMethod
Attached to: tfamily module:Cooling_Radii
File: tfamily cooling.cooling_radius.F90
Default value: simple
Description: The name of the method to be used for calculations of the cooling radius.

Name: tfamily coolingRateMethod
Attached to: tfamily module:Cooling_Rates
File: tfamily cooling.cooling_rate.F90
Default value: White-Frenk1991
Description: The name of the method to be used when computing the cooling rate.
4. Input Parameters

Name: tfamily coolingRateSimpleScalingCutOffExponent
Attached to: tfamily module:Cooling_Rates_Simple_Scaling
File: tfamily cooling.cooling_rate.simple_scaling.F90
Default value: 1
Description: The exponent appearing in the exponential term for cooling timescale in the simple scaling cooling rate model.

Name: tfamily coolingRateSimpleScalingCutOffMass
Attached to: tfamily module:Cooling_Rates_Simple_Scaling
File: tfamily cooling.cooling_rate.simple_scaling.F90
Default value: $10^{12} M_\odot$
Description: The halo mass scale appearing in the exponential term for cooling timescale in the simple scaling cooling rate model.

Name: tfamily coolingRateSimpleScalingCutOffWidth
Attached to: tfamily module:Cooling_Rates_Simple_Scaling
File: tfamily cooling.cooling_rate.simple_scaling.F90
Default value: 1
Description: The width appearing in the exponential term for cooling timescale in the simple scaling cooling rate model.

Name: tfamily coolingRateSimpleScalingTimescale
Attached to: tfamily module:Cooling_Rates_Simple_Scaling
File: tfamily cooling.cooling_rate.simple_scaling.F90
Default value: 1 Gyr
Description: The timescale (in Gyr) for cooling in low mass halos at $z = 0$ in the simple scaling cooling rate model.

Name: tfamily coolingRateSimpleScalingTimescaleExponent
Attached to: tfamily module:Cooling_Rates_Simple_Scaling
File: tfamily cooling.cooling_rate.simple_scaling.F90
Default value: $-1.5$
Description: The exponent of $(1 + z)$ in the cooling timescale for low mass halos in the simple scaling cooling rate model.

Name: tfamily coolingRateSimpleTimescale
Attached to: tfamily module:Cooling_Rates_Simple
File: tfamily cooling.cooling_rate.simple.F90
Default value: 1 Gyr
Description: The timescale (in Gyr) for cooling in the simple cooling rate model.

Name: tfamily coolingRotationVelocityFrom
Attached to: tfamily module:Cooling_Specific_Angular_Momenta_Constant_Rotation
File: tfamily cooling.specific_angular_momentum.constant_rotation.F90
Default value: hotGas
Description: The component ("hotGas" or "darkMatter") from which the constant rotation speed should be computed for calculations of cooling gas specific angular momentum.
Name: tfamily coolingSpecificAngularMomentumMethod
Attached to: tfamily module: Cooling_Specific_Angular_Momenta
File: tfamily cooling.specific_angular_momentum.F90
Default value: constantRotation
Description: The name of the method to be used for calculations of the specific angular momentum of cooling gas.

Name: tfamily coolingTimeAvailableAgeFactor
Attached to: tfamily module: Cooling_Time_Available_White_Frenk
File: tfamily cooling.time_available.White-Frenk.F90
Default value: 0
Description: Interpolates (geometrically) between the age of the Universe and the halo dynamical time for the time available for cooling in the White-Frenk1991 method.

Name: tfamily coolingTimeAvailableMethod
Attached to: tfamily module: Cooling_Times_Available
File: tfamily cooling.time_available.F90
Default value: White-Frenk1991
Description: The name of the method to be used when computing the time available for cooling.

Name: tfamily coolingTimeMethod
Attached to: tfamily module: Cooling_Times
File: tfamily cooling.cooling_time.F90
Default value: simple
Description: The name of the method to be use for computing cooling times.

Name: tfamily coolingTimeSimpleDegreesOfFreedom
Attached to: tfamily module: Cooling_Times_Simple
File: tfamily cooling.cooling_time.simple.F90
Default value: 3
Description: Number of degrees of freedom to assume when computing the energy density of cooling gas in the "simple" cooling time module.

Name: tfamily cosmologicalMassVarianceMethod
Attached to: tfamily module: Power_Spectra
File: tfamily structure_formation.power_spectrum.F90
Default value: filteredPowerSpectrum
Description: Selects the method to be used for computing the cosmological mass variance.

Name: tfamily cosmologyFunctionsMethod
Attached to: tfamily module: Cosmology_Functions
File: tfamily cosmology.functions.F90
Default value: matterLambda
Description: The method to be used for cosmologyFunctions.

Name: tfamily cosmologyParametersMethod
Attached to: tfamily module: Cosmology_Parameters
File: tfamily cosmology.parameters.F90
Default value: simple
4. Input Parameters

Description: The method to be used for cosmologyParameters.

Name: tfamily criticalOverdensityMassScalingMethod
Attached to: tfamily module:Critical_Overdensity
File: tfamily structure_formation.critical_overdensity.F90
Default value: null
Description: The name of the method to be used for scaling critical overdensities for halo collapse with mass.

Name: tfamily criticalOverdensityMethod
Attached to: tfamily module:Critical_Overdensity
File: tfamily structure_formation.critical_overdensity.F90
Default value: sphericalTopHat
Description: The name of the method to be used for critical overdensities for halo collapse.

Name: tfamily darkEnergyEquationOfStateW0
Attached to: tfamily module:Cosmology_Functions
File: tfamily cosmology.functions.F90
Default value: -1 (cosmological constant)
Description: The equation of state parameter for dark energy, \( w_0 \), defined such that \( P = \rho w \) with \( w(a) = w_0 + w_1 a(1 - a) \).

Name: tfamily darkEnergyEquationOfStateW1
Attached to: tfamily module:Cosmology_Functions
File: tfamily cosmology.functions.F90
Default value: 0 (constant equation of state)
Description: The equation of state parameter for dark energy, \( w_1 \), defined such that \( P = \rho w \) with \( w(a) = w_0 + w_1 a(1 - a) \).

Name: tfamily darkMatterAccretionHistoryMethod
Attached to: tfamily module:Dark_Matter_Halo_Mass_Accretion_Histories
File: tfamily dark_matter_halos.mass_accretion_history.F90
Default value: Wechsler2002
Description: The name of the method to be used for calculations of dark matter halo mass accretion histories.

Name: tfamily darkMatterHaloBiasMethod
Attached to: tfamily module:Dark_Matter_Halo_Biases
File: tfamily structure_formation.halo_bias.F90
Default value: Tinker2010
Description: Selects which dark matter halo bias method to use.

Name: tfamily darkMatterHaloMassLossRateMethod
Attached to: tfamily module:Dark_Matter_Halos_Mass_Loss_Rates
File: tfamily dark_matter_halos.mass_loss_rates.F90
Default value: dynamicalTime
Description: The name of the method to be used for computing mass loss rates from dark matter halos.

Name: tfamily darkMatterHaloScaleMethod
Attached to: tfamily module:Dark Matter Halo Scales
File: tfamily dark_matter_halos.scales.F90
Default value: virialDensityContrastDefinition
Description: The method to be used for darkMatterHaloScale.

Name: tfamily darkMatterProfileConcentrationCDMMethod
Attached to: tfamily module:Dark Matter Profiles Concentration
File: tfamily dark_matter_profiles.structure.concentration.F90
Default value: gao2008
Description: The darkMatterProfileConcentration method to which the Schneider et al. [2012] modifier for WDM halo concentrations should be applied.

Name: tfamily darkMatterProfileConcentrationDiemerKravtsov2014Alpha
Attached to: tfamily module:Dark Matter Profiles Concentration
File: tfamily dark_matter_profiles.structure.concentration.F90
Default value: 1.12
Description: The parameter \( \alpha \) appearing in the halo concentration algorithm of Diemer and Kravtsov [2014].

Name: tfamily darkMatterProfileConcentrationDiemerKravtsov2014Beta
Attached to: tfamily module:Dark Matter Profiles Concentration
File: tfamily dark_matter_profiles.structure.concentration.F90
Default value: 1.69
Description: The parameter \( \beta \) appearing in the halo concentration algorithm of Diemer and Kravtsov [2014].

Name: tfamily darkMatterProfileConcentrationDiemerKravtsov2014Eta0
Attached to: tfamily module:Dark Matter Profiles Concentration
File: tfamily dark_matter_profiles.structure.concentration.F90
Default value: 6.82
Description: The parameter \( \eta_0 \) appearing in the halo concentration algorithm of Diemer and Kravtsov [2014].

Name: tfamily darkMatterProfileConcentrationDiemerKravtsov2014Eta1
Attached to: tfamily module:Dark Matter Profiles Concentration
File: tfamily dark_matter_profiles.structure.concentration.F90
Default value: 1.42
Description: The parameter \( \eta_1 \) appearing in the halo concentration algorithm of Diemer and Kravtsov [2014].

Name: tfamily darkMatterProfileConcentrationDiemerKravtsov2014Kappa
Attached to: tfamily module:Dark Matter Profiles Concentration
File: tfamily dark_matter_profiles.structure.concentration.F90
Default value: 0.69
Description: The parameter \( \kappa \) appearing in the halo concentration algorithm of Diemer and Kravtsov [2014].

Name: tfamily darkMatterProfileConcentrationDiemerKravtsov2014Phi0
Attached to: tfamily module:Dark Matter Profiles Concentration
4. Input Parameters

File: tfamily dark_matter_profiles.structure.concentration.F90
Default value: 6.58
Description: The parameter $\phi_0$ appearing in the halo concentration algorithm of Diemer and Kravtsov [2014].

Name: tfamily darkMatterProfileConcentrationDiemerKravtsov2014Phi1
Attached to: tfamily module:Dark_Matter_Profiles_Concentration
File: tfamily dark_matter_profiles.structure.concentration.F90
Default value: 1.37
Description: The parameter $\phi_1$ appearing in the halo concentration algorithm of Diemer and Kravtsov [2014].

Name: tfamily darkMatterProfileConcentrationMethod
Attached to: tfamily module:Dark_Matter_Profiles_Concentration
File: tfamily dark_matter_profiles.structure.concentration.F90
Default value: gao2008
Description: The method to be used for darkMatterProfileConcentration.

Name: tfamily darkMatterProfileMaximumConcentration
Attached to: tfamily objects.nodes.components.dark_matter_profile.scale.F90
Default value: 100
Description: The maximum concentration allowed for dark matter profiles.

Name: tfamily darkMatterProfileMethod
Attached to: tfamily module:Dark_Matter_Profiles
File: tfamily dark_matter_profiles.F90
Default value: NFW
Description: The method to be used for darkMatterProfile.

Name: tfamily darkMatterProfileMinimumConcentration
Attached to: tfamily objects.nodes.components.dark_matter_profile.scale.F90
Default value: 4
Description: The minimum concentration allowed for dark matter profiles.

Name: tfamily darkMatterProfileScaleCorrectForConcentrationDefinition
Attached to: tfamily module:Dark_Matter_Profile_Scales
File: tfamily dark_matter_profiles.structure.scale.F90
Default value: false
Description: If true, then when computing dark matter profile scale radii using concentrations, any difference between the current definition of halo scales (i.e. typically virial density contrast definitions) and density profiles and those assumed in measuring the concentrations will be taken into account. If false, the concentration is applied blindly.

Name: tfamily darkMatterShapeMethod
Attached to: tfamily module:Dark_Matter_Profiles_Shapes
File: tfamily dark_matter_profiles.structure.shape.F90
Default value: Gao2008
**Description:** The name of the method to be used for calculations of dark matter halo density profile shapes.

**Name:** tfamily deltaFunctionSpinDistributionSpin  
**Attached to:** tfamily module:Halo_Spin_Distributions_Delta_Function  
**File:** tfamily dark_matter_halos.spins.distributions.delta_function.F90  
**Default value:** 0.03687 [Bett et al., 2007]  
**Description:** The fixed value of spin in a delta function spin distribution.

**Name:** tfamily dielectronicRecombinationMethod  
**Attached to:** tfamily module:Atomic_Rates_Recombination_Dielectronic  
**File:** tfamily atomic.rates.recombination.dielectronic.F90  
**Default value:** Arnaud85  
**Description:** The name of the method to be used for computing dielectronic recombination rates.

**Name:** tfamily diskMassToleranceAbsolute  
**Attached to:** tfamily module:Node_Component_Disk_Exponential  
**File:** tfamily objects.nodes.components.disk.exponential.F90  
**Default value:** $10^{-6} M_\odot$  
**Description:** The mass tolerance used to judge whether the disk is physically plausible.

**Name:** tfamily diskNegativeAngularMomentumAllowed  
**Attached to:** tfamily module:Node_Component_Disk_Exponential  
**File:** tfamily objects.nodes.components.disk.exponential.F90  
**Default value:** true  
**Description:** Specifies whether or not negative angular momentum is allowed for the disk.

**Name:** tfamily diskOutflowExponent  
**Attached to:** tfamily module:Star_Formation_Feedback_Disks_Power_Law  
**File:** tfamily star_formation.feedback.disks.power_law.F90  
**Default value:** 3.5  
**Description:** The velocity scaling of the supernovae (SNe)-driven outflow rate in disks.

**Name:** tfamily diskOutflowFraction  
**Attached to:** tfamily module:Star_Formation_Feedback_Disks_Halo_Scaling  
**File:** tfamily star_formation.feedback.disks.halo_scaling.F90  
**Default value:** 0.01  
**Description:** The ratio of outflow rate to star formation rate in disks.

**Name:** tfamily diskOutflowRedshiftExponent  
**Attached to:** tfamily module:Star_Formation_Feedback_Disks_Halo_Scaling  
**File:** tfamily star_formation.feedback.disks.halo_scaling.F90  
**Default value:** 0.0  
**Description:** The exponent of redshift in the outflow rate in disks.

**Name:** tfamily diskOutflowTimescaleMinimum  
**Attached to:** tfamily module:Node_Component_Disk_Exponential  
**File:** tfamily objects.nodes.components.disk.exponential.F90  
**Default value:** 10^{-3}
4. Input Parameters

**Description:** The minimum timescale (in units of the disk dynamical time) on which outflows may deplete gas in the disk.

**Name:** tfamily diskOutflowVelocity
**Attached to:** tfamily module:Star_Formation_Feedback_Disks_Power_Law
**File:** tfamily star_formation.feedback.disks.power_law.F90
**Default value:** 250
**Description:** The velocity scale at which the SNe-driven outflow rate equals the star formation rate in disks.

**Name:** tfamily diskOutflowVirialVelocityExponent
**Attached to:** tfamily module:Star_Formation_Feedback_Disks_Halo_Scaling
**File:** tfamily star_formation.feedback.disks.halo_scaling.F90
**Default value:** $-2.0$
**Description:** The exponent of virial velocity in the outflow rate in disks.

**Name:** tfamily diskOutputStarFormationRate
**Attached to:** tfamily module:Galacticus_Nodes
**File:** tfamily objects.nodes.F90
**Default value:** false
**Description:** Specifies whether the `starFormationRate` method of the exponential implementation of the disk component class should be output.

**Name:** tfamily diskRadiusSolverCole2000Method
**Attached to:** tfamily module:Node_Component_Disk_Exponential
**File:** tfamily objects.nodes.components.disk.exponential.F90
**Default value:** 1
**Description:** HASH(0x157f9e8)

**Name:** tfamily diskStarFormationTimescaleMinimum
**Attached to:** tfamily module:Node_Component_Disk_Very_Simple
**File:** tfamily objects.nodes.components.disk.very_simple.F90
**Default value:** $10^{-3}$
**Description:** The minimum timescale (in units of the halo dynamical time) on which star formation may occur in the disk.

**Name:** tfamily diskStructureSolverRadius
**Attached to:** tfamily module:Node_Component_Disk_Exponential
**File:** tfamily objects.nodes.components.disk.exponential.F90
**Default value:** 1
**Description:** The radius (in units of the exponential scale length) to use in solving for the size of the disk.

**Name:** tfamily diskSuperwindMassLoading
**Attached to:** tfamily module:Star_Formation_Expulsive_Feedback_Disks_Superwind
**File:** tfamily star_formation.feedback.expulsion.disks.superwind.F90
**Default value:** 2
**Description:** The mass loading of the disk superwind.

**Name:** tfamily diskSuperwindVelocity
Attached to: tfamily module:Star_Formation_Expulsive_Feedback_Disks_Superwind
File: tfamily star_formation.feedback.expulsion.disks.superwind.F90
Default value: 200 km/s
Description: The velocity scale of the disk superwind.

Name: tfamily duttonMaccio2014FitType
Attached to: tfamily module:Dark_Matter_Profiles_Concentration
File: tfamily dark_matter_profiles.structure.concentration.F90
Default value: nfwVirial
Description: The parameter $A$ appearing in the halo concentration algorithm of Prada et al. [2011].

Name: tfamily dynamicsStatisticsBarsFrequency
Attached to: tfamily module:Node_Component_Dynamics_Statistics_Bars
File: tfamily objects.nodes.components.dynamics_statistics.bars.F90
Default value: 0.1
Description: The frequency (in fractions of the host halo dynamical time) at which to record the bar dynamical status of satellite galaxies.

Name: tfamily effectiveNumberNeutrinos
Attached to: tfamily module:Transfer_Function_Eisenstein_Hu
File: tfamily structure_formation.transfer_function.Eisenstein_Hu.F90
Default value: 3.046 [Mangano et al., 2005]
Description: The effective number of neutrino species as used in the Eisenstein and Hu [1999] transfer function.

Name: tfamily elementsToTrack
Attached to: tfamily module:Abundances_Structure
File: tfamily objects.abundances.F90
Default value: none
Description: The names of the elements to be tracked.

Name: tfamily excursionSetBarrierConstantCoefficient
Attached to: tfamily module:Excursion_Sets_Barriers_Quadratic
File: tfamily structure_formation.excursion_sets.barrier.quadratic.F90
Default value: 1.67
Description: The constant term in the excursion set barrier.

Name: tfamily excursionSetBarrierLinearCoefficient
Attached to: tfamily module:Excursion_Sets_Barriers_Quadratic
File: tfamily structure_formation.excursion_sets.barrier.quadratic.F90
Default value: 0.0
Description: The coefficient of the linear term in the excursion set barrier.

Name: tfamily excursionSetBarrierMethod
Attached to: tfamily module:Excursion_Sets_Barriers
File: tfamily structure_formation.excursion_sets.barrier.F90
Default value: criticalOverdensity
Description: The name of the method to be used for calculations of excursion set barriers.
4. Input Parameters

Name: tfamily excursionSetBarrierQuadraticCoefficient
Attached to: tfamily module:Excursion_Sets_Barriers_Quadratic
File: tfamily structure_formation.excursion_sets.barrier.quadratic.F90
Default value: 0.0
Description: The coefficient of the quadratic term in the excursion set barrier.

Name: tfamily excursionSetBarrierRatesRemapMethods
Attached to: tfamily module:Excursion_Sets_Barriers
File: tfamily structure_formation.excursion_sets.barrier.F90
Default value: null
Description: The name of the method to be used for remapping excursion set barriers for rate calculations.

Name: tfamily excursionSetBarrierRemapMethods
Attached to: tfamily module:Excursion_Sets_Barriers
File: tfamily structure_formation.excursion_sets.barrier.F90
Default value: null
Description: The name of the method to be used for remapping excursion set barriers.

Name: tfamily excursionSetBarrierRemapScalingFactor
Attached to: tfamily module:Excursion_Sets_Barriers_Remap_Scale
File: tfamily structure_formation.excursion_sets.barrier.remap.scale.F90
Default value: 1
Description: The factor by which the excursion set barrier should be rescaled if the scale remapping method is active.

Name: tfamily excursionSetFirstCrossingFarahiFileName
Attached to: tfamily module:Excursion_Sets_First_Crossing_Farahi
File: tfamily structure_formation.excursion_sets.first_crossing_distribution.Farahi.F90
Default value: none
Description: The name of the file to/from which tabulations of barrier first crossing probabilities should be written/read. If set to "none" tables will not be stored.

Name: tfamily excursionSetFirstCrossingFarahiFractionalTimeStep
Attached to: tfamily module:Excursion_Sets_First_Crossing_Farahi
File: tfamily structure_formation.excursion_sets.first_crossing_distribution.Farahi.F90
Default value: 0.01
Description: The fractional time step used when computing barrier crossing rates in the Farahi excursion set solver (i.e. the step used in finite difference calculations).

Name: tfamily excursionSetFirstCrossingMethod
Attached to: tfamily module:Excursion_Sets_First_Crossings
File: tfamily structure_formation.excursion_sets.first_crossing_distribution.F90
Default value: linearBarrier
Description: The name of the method to be used for calculations of first crossing distributions for excursion sets.

Name: tfamily filterLightconeFixedTime
Attached to: tfamily module:Galacticus_Merger_Tree_Output_Filter_Lightcones
File: tfamily galacticus.output.merger_tree.filters.lightcone.F90
Default value: false
Description: Specifies if lightcone output should occur at a fixed time (as opposed to the usual case where time evolves along the lightcone). Intended for the construction of lightcones with no evolution.

Name: tfamily filterLightconeGeometryFileName
Attached to: tfamily module: Galacticus_Merger_Tree_Output_Filter_Lightcones
File: tfamily galacticus.output.merger_tree.filters.lightcone.F90
Default value: none
Description: The name of an XML file from which to read details of lightcone geometry.

Name: tfamily filterLightconePruneTrees
Attached to: tfamily module: Galacticus_Merger_Tree_Output_Filter_Lightcones
File: tfamily galacticus.output.merger_tree.filters.lightcone.F90
Default value: none
Description: Specifies whether trees which lie wholly outside of the lightcone geometry should be pruned.

Name: tfamily freefallRadiusMethod
Attached to: tfamily module: Freefall_Radii
File: tfamily cooling.freefall_radii.F90
Default value: darkMatterHalo
Description: The name of the method to be used for calculations of the freefall radius in cooling calculations.

Name: tfamily freefallTimeAvailableMethod
Attached to: tfamily module: Cooling_Freefall_Times_Available
File: tfamily cooling.freefall_time_available.F90
Default value: haloFormation
Description: The name of the method to be used when computing the time available for freefall in cooling calculations.

Name: tfamily galacticStructureRadiiFixedFactor
Attached to: tfamily module: Galactic_Structure_Radii_Fixed
File: tfamily galactic_structure.radius_solver.fixed.F90
Default value: $\sqrt{1/2} \ [\text{Mo et al., 1998}]$
Description: The ratio of galaxy radius to $\lambda_{\text{vir}}$ in the “fixed” galactic structure radius solver algorithm.

Name: tfamily galacticStructureRadiusSolverInitialRadiusMethod
Attached to: tfamily module: Galactic_Structure_Initial_Radii
File: tfamily galactic_structure.radius_solver.initial_radii.F90
Default value: adiabatic
Description: Selects the method to be used to determine initial radii in the dark matter halo when solving for galactic structure.

Name: tfamily galacticStructureRadiusSolverMethod
Attached to: tfamily module: Galactic_Structure_Radii
File: tfamily galactic_structure.radius_solver.F90
Default value: adiabatic
Description: Selects the method to be used for solving for galactic structure.
4. Input Parameters

Name: tfamily galacticusOutputFileName
Attached to: tfamily module:Galacticus_Output_Open
File: tfamily galacticus.output.HDF5.open.F90
Default value: galacticus.hdf5
Description: The name of the file to which GALACTICUS results will be written.

Name: tfamily galacticusOutputScratchFileName
Attached to: tfamily module:Galacticus_Output_Open
File: tfamily galacticus.output.HDF5.open.F90
Default value: [galacticusOutputFileName]
Description: The name of the file to which GALACTICUS results will be written temporarily during runs.

Name: tfamily gauntFactorMethod
Attached to: tfamily module:Atomic_Radiation_Gaunt_Factors
File: tfamily atomic.radiation.gaunt_factors.F90
Default value: sutherland1998
Description: The name of the method to be used for computing Gaunt factors.

Name: tfamily gaussianRandomSeed
Attached to: tfamily module:Gaussian_Random
File: tfamily numerical.random.gaussian.F90
Default value: 843
Description: A seed for the Gaussian random number generator.

Name: tfamily generalizedPressSchechterDeltaStepMaximum
Attached to: tfamily module:Generalized_Press_Schechter_Branching
File: tfamily merger_trees.branching_probability.generalized_Press_Schechter.F90
Default value: 0.1
Description: Limits the step in $\delta_{\text{crit}}$ when constructing merger trees using the generalized Press-Schechter branching algorithm.

Name: tfamily generalizedPressSchechterMinimumMass
Attached to: tfamily module:Generalized_Press_Schechter_Branching
File: tfamily merger_trees.branching_probability.generalized_Press_Schechter.F90
Default value: 1.0d6
Description: The minimum mass to used in computing subresolution accretion rates when constructing merger trees using the generalized Press-Schechter branching algorithm.

Name: tfamily generalizedPressSchechterSmoothAccretion
Attached to: tfamily module:Generalized_Press_Schechter_Branching
File: tfamily merger_trees.branching_probability.generalized_Press_Schechter.F90
Default value: true
Description: Specifies whether or not to include smooth accretion in subresolution accretion rates when constructing merger trees using the generalized Press-Schechter branching algorithm.

Name: tfamily haloMassFunctionMethod
Attached to: tfamily module:Halo_Mass_Function
File: tfamily structure_formation.halo_mass_function.F90
**Default value:** Tinker2008  
**Description:** The name of the method to be used for computing the dark matter halo mass function.

**Name:** tfamily haloMassFunctionSamplingAbundanceMaximum  
**Attached to:** tfamily module:Merger_Trees_Mass_Function_Sampling_Halo_MF  
**File:** tfamily merger_trees.construct.mass_function_sampling.halo_mass_function.F90  
**Default value:** -1  
**Description:** The abundance (in units of Mpc$^{-3}$) above which to truncate the halo mass function when sampling halo masses for tree construction. A negative value indicates no truncation.

**Name:** tfamily haloMassFunctionSamplingAbundanceMinimum  
**Attached to:** tfamily module:Merger_Trees_Mass_Function_Sampling_Halo_MF  
**File:** tfamily merger_trees.construct.mass_function_sampling.halo_mass_function.F90  
**Default value:** -1  
**Description:** The abundance (in units of Mpc$^{-3}$) below which to truncate the halo mass function when sampling halo masses for tree construction. A negative value indicates no truncation.

**Name:** tfamily haloMassFunctionSamplingMethod  
**Attached to:** tfamily module:Merger_Trees_Mass_Function_Sampling  
**File:** tfamily merger_trees.construct.mass_function_sampling.F90  
**Default value:** haloMassFunction  
**Description:** The name of the method to be used for sampling the halo mass function when constructing merger trees.

**Name:** tfamily haloMassFunctionSamplingStellarMassFunctionErrorAlpha  
**Attached to:** tfamily module:Merger_Trees_Mass_Function_Sampling_Stellar_MF  
**File:** tfamily merger_trees.construct.mass_function_sampling.stellar_mass_function.F90  
**Default value:** none  
**Description:** The value $\alpha$ in a Schechter function describing the errors on the stellar mass function to be assumed when computing the optimal sampling density function for tree masses.

**Name:** tfamily haloMassFunctionSamplingStellarMassFunctionErrorBeta  
**Attached to:** tfamily module:Merger_Trees_Mass_Function_Sampling_Stellar_MF  
**File:** tfamily merger_trees.construct.mass_function_sampling.stellar_mass_function.F90  
**Default value:** none  
**Description:** The value $\beta$ in a Schechter function describing the errors on the stellar mass function to be assumed when computing the optimal sampling density function for tree masses.

**Name:** tfamily haloMassFunctionSamplingStellarMassFunctionErrorConstant  
**Attached to:** tfamily module:Merger_Trees_Mass_Function_Sampling_Stellar_MF  
**File:** tfamily merger_trees.construct.mass_function_sampling.stellar_mass_function.F90  
**Default value:** none  
**Description:** The constant error contribution to the stellar mass function to be assumed when computing the optimal sampling density function for tree masses.

**Name:** tfamily haloMassFunctionSamplingStellarMassFunctionErrorLogBinWidth  
**Attached to:** tfamily module:Merger_Trees_Mass_Function_Sampling_Stellar_MF  
**File:** tfamily merger_trees.construct.mass_function_sampling.stellar_mass_function.F90  
**Default value:** none
4. Input Parameters

**Description:** The logarithmic width of bins in the stellar mass function to be assumed when computing the optimal sampling density function for tree masses.

**Name:** tfamily haloMassFunction_SamplingStellarMassFunction_ErrorMassMaximum
**Attached to:** tfamily module:Merger_Trees_Mass_Function_Sampling_Stellar_MF
**File:** tfamily merger_trees.construct.mass_function_sampling.stellar_mass_function.F90
**Default value:** none
**Description:** The minimum stellar mass to consider when computing the optimal sampling density function for tree masses.

**Name:** tfamily haloMassFunction_SamplingStellarMassFunction_ErrorMassMinimum
**Attached to:** tfamily module:Merger_Trees_Mass_Function_Sampling_Stellar_MF
**File:** tfamily merger_trees.construct.mass_function_sampling.stellar_mass_function.F90
**Default value:** none
**Description:** The minimum stellar mass to consider when computing the optimal sampling density function for tree masses.

**Name:** tfamily haloMassFunction_SamplingStellarMassFunction_ErrorMstar
**Attached to:** tfamily module:Merger_Trees_Mass_Function_Sampling_Stellar_MF
**File:** tfamily merger_trees.construct.mass_function_sampling.stellar_mass_function.F90
**Default value:** none
**Description:** The value $M_\star$ in a Schechter function describing the errors on the stellar mass function to be assumed when computing the optimal sampling density function for tree masses.

**Name:** tfamily haloMassFunction_SamplingStellarMassFunction_ErrorPhi0
**Attached to:** tfamily module:Merger_Trees_Mass_Function_Sampling_Stellar_MF
**File:** tfamily merger_trees.construct.mass_function_sampling.stellar_mass_function.F90
**Default value:** none
**Description:** The value $\phi_0$ in a Schechter function, $\sigma(M) = \phi_0(M/M_\star)^\alpha \exp(-[M/M_\star]^{\beta})$, describing the errors on the stellar mass function to be assumed when computing the optimal sampling density function for tree masses.

**Name:** tfamily haloMassFunction_SimpleSystematicAlpha
**Attached to:** tfamily module:Halo_Mass_FUNCTION_Modifiers_Simple_Systematic
**File:** tfamily structure_formation.halo_mass_function.modifier.simple_systematic.F90
**Default value:** 0
**Description:** Parameter $\alpha$ appearing in model for simple systematic shift in the halo mass function.

**Name:** tfamily haloMassFunction_SimpleSystematicBeta
**Attached to:** tfamily module:Halo_Mass_FUNCTION_Modifiers_Simple_Systematic
**File:** tfamily structure_formation.halo_mass_function.modifier.simple_systematic.F90
**Default value:** 0
**Description:** Parameter $\beta$ appearing in model for simple systematic shift in the halo mass function.

**Name:** tfamily haloMassFunctionsMassMaximum
**Attached to:** tfamily module:Halo_Mass_Function_Tasks
**File:** tfamily halo_mass_functions.tasks.F90
**Default value:** $10^{15}$
**Description:** The maximum mass at which to tabulate halo mass functions.
Name: `tfamily haloMassFunctionsMassMinimum`  
**Attached to:** `tfamily module:Halo_MassFunction_Tasks`  
**File:** `tfamily halo_mass_functions.tasks.F90`  
**Default value:** $10^{10}$  
**Description:** The minimum mass at which to tabulate halo mass functions.

Name: `tfamily haloMassFunctionsPointsPerDecade`  
**Attached to:** `tfamily module:Halo_MassFunction_Tasks`  
**File:** `tfamily halo_mass_functions.tasks.F90`  
**Default value:** 10  
**Description:** The number of points per decade of halo mass at which to tabulate halo mass functions.

Name: `tfamily haloMassesIncludeSubhalos`  
**Attached to:** `tfamily module:Merger_Trees_Simple`  
**File:** `tfamily merger_trees.file_maker.simple.F90`  
**Default value:** none  
**Description:** Specifies whether or not halo masses include the masses of their subhalos.

Name: `tfamily haloModelWavenumberMaximum`  
**Attached to:** `tfamily module:Galacticus_Output_Halo_Models`  
**File:** `tfamily galacticus.output.merger_tree.halo_model.F90`  
**Default value:** $10^4$  
**Description:** The maximum wavenumber (in Mpc$^{-1}$) at which to tabulate power spectra for the halo model.

Name: `tfamily haloModelWavenumberMinimum`  
**Attached to:** `tfamily module:Galacticus_Output_Halo_Models`  
**File:** `tfamily galacticus.output.merger_tree.halo_model.F90`  
**Default value:** $10^{-3}$  
**Description:** The minimum wavenumber (in Mpc$^{-1}$) at which to tabulate power spectra for the halo model.

Name: `tfamily haloModelWavenumberPointsPerDecade`  
**Attached to:** `tfamily module:Galacticus_Output_Halo_Models`  
**File:** `tfamily galacticus.output.merger_tree.halo_model.F90`  
**Default value:** 10  
**Description:** The number of points per decade in wavenumber at which to tabulate power spectra for the halo model.

Name: `tfamily haloReformationMassFactor`  
**Attached to:** `tfamily module:Node_Component_Formation_Times_Cole2000`  
**File:** `tfamily objects.nodes.components.formation_times.Cole2000.F90`  
**Default value:** 2.0  
**Description:** Factor by which halo mass must have increased to trigger a new formation event.

Name: `tfamily haloReformationOnPromotionOnly`  
**Attached to:** `tfamily module:Node_Component_Formation_Times_Cole2000`  
**File:** `tfamily objects.nodes.components.formation_times.Cole2000.F90`
4. Input Parameters

Default value: false
Description: Specifies whether halo reformation should occur only at node promotion events, or at the precise time that the halo mass has increased sufficiently in mass.

Name: tfamily haloSpinDistributionMethod
Attached to: tfamily module:Halo_Spin_Distributions
File: tfamily dark_matter_halos.spins.distributions.F90
Default value: Bett2007
Description: The name of the method to be use for computing halo spin distributions.

Name: tfamily harrassmentMassThreshold
Attached to: tfamily module:Galactic_Dynamics_Bar_Instabilities_ELN_Tidal
File: tfamily galactic_dynamics.bar_instability.ELN_tidal.F90
Default value: 0.0
Description: The host halo mass threshold for harrassment to take effect.

Name: tfamily hdf5CacheElementsCount
Attached to: tfamily module:Galacticus_Output_Open
File: tfamily galacticus.output.HDF5.open.F90
Default value: 521
Description: Number of elements in the raw data chunk cache.

Name: tfamily hdf5CacheSizeBytes
Attached to: tfamily module:Galacticus_Output_Open
File: tfamily galacticus.output.HDF5.open.F90
Default value: 1048576
Description: Size of the raw data chunk cache in bytes.

Name: tfamily hdf5ChunkSize
Attached to: tfamily module:Galacticus_Output_Open
File: tfamily galacticus.output.HDF5.open.F90
Default value: 1024
Description: The chunk size used for outputting HDF5 datasets.

Name: tfamily hdf5CompressionLevel
Attached to: tfamily module:Galacticus_Output_Open
File: tfamily galacticus.output.HDF5.open.F90
Default value: -1
Description: The compression level used for outputting HDF5 datasets.

Name: tfamily hdf5SieveBufferSize
Attached to: tfamily module:Galacticus_Output_Open
File: tfamily galacticus.output.HDF5.open.F90
Default value: 65536
Description: The size of the sieve buffer used by the HDF5 library to speed reading/writing of partial datasets.

Name: tfamily hdf5UseLatestFormat
Attached to: tfamily module:Galacticus_Output_Open
File: tfamily galacticus.output.HDF5.open.F90
Default value: false
Description: Specifies whether to use the latest HDF5 file format.

Name: tfamily heightToRadialScaleDisk
Attached to: tfamily module:Node_Component_Disk_Exponential
File: tfamily objects.nodes.components.disk.exponential.F90
Default value: 0.137 [Kregel et al., 2002]
Description: The ratio of scale height to scale radius for exponential disks.

Name: tfamily heightToRadialScaleDiskBlitzRosolowsky
Attached to: tfamily module:Star(Formation Rate_Surface_Density_Disks_BR
File: tfamily star_formation.rate_surface_density.disks.Blitz-Rosolowsky.F90
Default value: 0.137 [Kregel et al., 2002]
Description: The ratio of scale height to scale radius for disks in the "Blitz-Rosolowsky" star formation timescale calculation.

Name: tfamily hotHaloAngularMomentumAlwaysGrows
Attached to: tfamily module:Node_Component_Hot_Halo_Standard
File: tfamily objects.nodes.components.hot_halo.standard.F90
Default value: false
Description: Specifies whether or not negative rates of accretion of angular momentum into the hot halo will be treated as positive for the purposes of computing the hot halo angular momentum.

Name: tfamily hotHaloAngularMomentumLossFraction
Attached to: tfamily module:Node_Component_Hot_Halo_Standard
File: tfamily objects.nodes.components.hot_halo.standard.F90
Default value: 0.3
Description: Specifies the fraction of angular momentum that is lost from cooling/infalling gas.

Name: tfamily hotHaloColdModeCoredIsothermalCoreRadiiMethod
Attached to: tfamily module:Hot_Halo_Cold_Mode_Density_Cored_Isothermal_Core_Radii
File: tfamily hot_halo.cold_mode.density_profile.cored_isothermal.core_radius.F90
Default value: virialRadiusFraction
Description: The name of the method to be used for computing the core radii of cored isothermal cold mode hot halo profiles.

Name: tfamily hotHaloCoolingFromNode
Attached to: tfamily module:Node_Component_Hot_Halo_Standard
File: tfamily objects.nodes.components.hot_halo.standard.F90
Default value: currentNode
Description: Specifies whether the angular momentum of cooling gas should be computed from the "current node" or the "formation node".

Name: tfamily hotHaloCoreRadiusOverScaleRadius
Attached to: tfamily function:growingDefaultConstructor
File: tfamily hot_halo.mass_distribution.cored.core_radius.F90
Default value: 0.1
Description: The core radius in the hot halo density profile in units of the dark matter profile scale
4. Input Parameters

Name: tfamily hotHaloCoreRadiusOverVirialRadius
Attached to: tfamily module:Hot_Halo_Mass_Distributions_CORE_Radii
File: tfamily hot_halo.mass_distribution.cored.core_radius.F90
Default value: 0.3
Description: The core radius in the hot halo density profile in units of the virial radius.

Name: tfamily hotHaloCoreRadiusOverVirialRadiusMaximum
Attached to: tfamily function: growingDefaultConstructor
File: tfamily hot_halo.mass_distribution.cored.core_radius.F90
Default value: 10
Description: The maximum core radius in the “cored isothermal” hot halo density profile in units of the virial radius.

Name: tfamily hotHaloExcessHeatDrivesOutflow
Attached to: tfamily module:Node_Component_Hot_Halo_Standard
File: tfamily objects.nodes.components.hot_halo.standard.F90
Default value: true
Description: Specifies whether heating of the halo in excess of its cooling rate will drive an outflow from the halo.

Name: tfamily hotHaloExpulsionRateMaximum
Attached to: tfamily module:Node_Component_Hot_Halo_Standard
File: tfamily objects.nodes.components.hot_halo.standard.F90
Default value: 1
Description: Specifies the maximum rate at which mass can be expelled from the hot halo in units of the inverse halo dynamical time.

Name: tfamily hotHaloMassDistributionBeta
Attached to: tfamily module:Hot_Halo_Mass_Distributions
File: tfamily hot_halo.mass_distribution.F90
Default value: 2/3
Description: The value of $\beta$ in $\beta$-profile hot halo mass distributions.

Name: tfamily hotHaloMassDistributionCoreRadiusMethod
Attached to: tfamily module:Hot_Halo_Mass_Distributions_CORE_Radii
File: tfamily hot_halo.mass_distribution.cored.core_radius.F90
Default value: virialFraction
Description: The method to be used for hotHaloMassDistributionCoreRadius.

Name: tfamily hotHaloMassDistributionMethod
Attached to: tfamily module:Hot_Halo_Mass_Distributions
File: tfamily hot_halo.mass_distribution.F90
Default value: betaProfile
Description: The method to be used for hotHaloMassDistribution.

Name: tfamily hotHaloNodeMergerLimitBaryonFraction
Attached to: tfamily module:Node_Component_Hot_Halo_Standard
File: tfamily objects.nodes.components.hot_halo.standard.F90
Default value: false
Description: Controls whether the hot gas content of nodes should be limited to not exceed the universal baryon fraction at node merger events. If set to true, hot gas (and angular momentum, abundances, and chemicals proportionally) will be removed from the merged halo to the unaccreted gas reservoir to limit the baryonic mass to the universal baryon fraction where possible.

Name: tfamily hotHaloOutflowReturnOnFormation
Attached to: tfamily module:Node_Component_Hot_Halo_Standard
File: tfamily objects.nodes.components.hot_halo.standard.F90
Default value: false
Description: Specifies whether or not outflowed gas should be returned to the hot reservoir on halo formation events.

Name: tfamily hotHaloOutflowReturnRate
Attached to: tfamily module:Node_Component_Hot_Halo_Standard
File: tfamily objects.nodes.components.hot_halo.standard.F90
Default value: 5
Description: Specifies the rate at which reheated mass is returned to the hot phase in units of the inverse halo dynamical time.

Name: tfamily hotHaloOutflowStrippingEfficiency
Attached to: tfamily module:Node_Component_Hot_Halo_Standard
File: tfamily objects.nodes.components.hot_halo.standard.F90
Default value: 0.1
Description: Specifies the efficiency with which outflowing gas is stripped from the hot halo, following the prescription of Font et al. (2008; i.e. this is the parameter $\epsilon_{\text{strip}}$ in their eqn. 6).

Name: tfamily hotHaloOutflowToColdMode
Attached to: tfamily module:Node_Component_Hot_Halo_Cold_Mode
File: tfamily objects.nodes.components.hot_halo.cold_mode.F90
Default value: false
Description: Specifies whether or not outflows from galaxies are returned to the cold or hot modes in the hot halo.

Name: tfamily hotHaloRamPressureForceMethod
Attached to: tfamily module:Hot_Halo_Ram_Pressure_Forces
File: tfamily hot_halo.ram_pressure_force.F90
Default value: Font2008
Description: The name of the method to be used when computing ram pressure force on hot halos.

Name: tfamily hotHaloRamPressureStrippingMethod
Attached to: tfamily module:Hot_Halo_Ram_Pressure_Stripping
File: tfamily hot_halo.ram_pressure_stripping.F90
Default value: Font2008
Description: The name of the method to be used when computing ram pressure stripping of hot halos.

Name: tfamily hotHaloRamPressureStrippingTimescaleMethod
Attached to: tfamily module:Hot_Halo_Ram_Pressure_Stripping_Timescales
4. Input Parameters

File: tfamily hot_halo.ram_pressure_stripping.timescale.F90
Default value: ramPressureAcceleration
Description: The name of the method to be used when computing ram pressure stripping timescales for hot halos.

Name: tfamily hotHaloTemperatureMethod
Attached to: tfamily module:Hot_Halo_Temperature_Profile
File: tfamily hot_halo.temperature_profile.F90
Default value: virial
Description: The name of the method to be used for computing hot halo temperature profiles.

Name: tfamily hotHaloTrackStrippedGas
Attached to: tfamily module:Node_Component_Hot_Halo_Standard
File: tfamily objects.nodes.components.hot_halo.standard.F90
Default value: true
Description: Specifies whether or not gas stripped from the hot halo should be tracked.

Name: tfamily hydrogenNetworkCMBOnly
Attached to: tfamily module:Chemical_Hydrogen_Rates
File: tfamily chemical.reaction_rates.hydrogen.F90
Default value: true
Description: Specifies whether or not to use the cosmic microwave background only when computed certain radiative rates.

Name: tfamily hydrogenNetworkFast
Attached to: tfamily module:Chemical_Hydrogen_Rates
File: tfamily chemical.reaction_rates.hydrogen.F90
Default value: true
Description: Specifies whether or not to use simplifying assumptions to speed the hydrogen network calculation. If true, H⁻ is assumed to be at equilibrium abundance, H₂⁺ reactions are ignored and other slow reactions are ignored (see Abel et al. 1997).

Name: tfamily igmPropertiesCompute
Attached to: tfamily module:Intergalactic_Medium_State_Internal_Evolver
File: tfamily intergalactic_medium.state.internal.evolver.F90
Default value: false
Description: Specifies whether or not the properties should be computed.

Name: tfamily igmPropertiesRedshiftMaximum
Attached to: tfamily module:Intergalactic_Medium_State_Internal_Evolver
File: tfamily intergalactic_medium.state.internal.evolver.F90
Default value: 400
Description: The maximum redshift to use in calculations.

Name: tfamily igmPropertiesRedshiftMinimum
Attached to: tfamily module:Intergalactic_Medium_State_Internal_Evolver
File: tfamily intergalactic_medium.state.internal.evolver.F90
Default value: 0
Description: The minimum redshift to use in calculations.
<table>
<thead>
<tr>
<th>Name: tfamily igmPropertiesTimeCountPerDecade</th>
<th>Attached to: tfamily module:Intergalactic_Medium_State_Internal_Evolver</th>
<th>File: tfamily intergalactic_medium.state.internal.evolver.F90</th>
<th>Default value: 10</th>
<th>Description: The number of bins per decade of time to use for calculations of the properties of the universe.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: tfamily igmStateSimplePreReionizationTemperature</td>
<td>Attached to: tfamily module:Intergalactic_Medium_State</td>
<td>File: tfamily intergalactic_medium.state.F90</td>
<td>Default value: 10K</td>
<td>Description: The pre-reionization temperature (in units of Kelvin) in the simple IGM state model.</td>
</tr>
<tr>
<td>Name: tfamily igmStateSimpleReionizationTemperature</td>
<td>Attached to: tfamily module:Intergalactic_Medium_State</td>
<td>File: tfamily intergalactic_medium.state.F90</td>
<td>Default value: $10^4$K</td>
<td>Description: The post-reionization temperature (in units of Kelvin) in the simple IGM state model.</td>
</tr>
<tr>
<td>Name: tfamily imfChabrierRecycledInstantaneous</td>
<td>Attached to: tfamily module:Star_Formation_IMF_Chabrier</td>
<td>File: tfamily star_formation.IMF.Chabrier.F90</td>
<td>Default value: 0.46 (internally computed)</td>
<td>Description: The recycled fraction for the Chabrier IMF in the instantaneous recycling approximation.</td>
</tr>
<tr>
<td>Name: tfamily imfChabrierYieldInstantaneous</td>
<td>Attached to: tfamily module:Star_Formation_IMF_Chabrier</td>
<td>File: tfamily star_formation.IMF.Chabrier.F90</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4. Input Parameters

**Default value:** 0.035 (internally computed)
**Description:** The yield for the Chabrier IMF in the instantaneous recycling approximation.

**Name:** tfamily imfKennicuttRecycledInstantaneous  
**Attached to:** tfamily module:Star_Formation_IMF_Kennicutt  
**File:** tfamily star_formation.IMF.Kennicutt.F90  
**Default value:** 0.57 (internally computed)  
**Description:** The recycled fraction for the Kennicutt IMF in the instantaneous recycling approximation.

**Name:** tfamily imfKennicuttYieldInstantaneous  
**Attached to:** tfamily module:Star_Formation_IMF_Kennicutt  
**File:** tfamily star_formation.IMF.Kennicutt.F90  
**Default value:** 0.044 (internally computed)  
**Description:** The yield for the Kennicutt IMF in the instantaneous recycling approximation.

**Name:** tfamily imfKroupaRecycledInstantaneous  
**Attached to:** tfamily module:Star_Formation_IMF_Kroupa  
**File:** tfamily star_formation.IMF.Kroupa.F90  
**Default value:** 0.30 (internally computed)  
**Description:** The recycled fraction for the Kroupa IMF in the instantaneous recycling approximation.

**Name:** tfamily imfKroupaYieldInstantaneous  
**Attached to:** tfamily module:Star_Formation_IMF_Kroupa  
**File:** tfamily star_formation.IMF.Kroupa.F90  
**Default value:** 0.023 (internally computed)  
**Description:** The yield for the Kroupa IMF in the instantaneous recycling approximation.

**Name:** tfamily imfMillerScaloRecycledInstantaneous  
**Attached to:** tfamily module:Star_Formation_IMF_MillerScalo  
**File:** tfamily star_formation.IMF.Miller-Scalo.F90  
**Default value:** 0.52 (computed internally)  
**Description:** The recycled fraction for the MillerScalo IMF in the instantaneous recycling approximation.

**Name:** tfamily imfMillerScaloYieldInstantaneous  
**Attached to:** tfamily module:Star_Formation_IMF_MillerScalo  
**File:** tfamily star_formation.IMF.Miller-Scalo.F90  
**Default value:** 0.026 (internally computed)  
**Description:** The yield for the MillerScalo IMF in the instantaneous recycling approximation.

**Name:** tfamily imfPiecewisePowerLawExponents  
**Attached to:** tfamily module:Star_Formation_IMF_PiecewisePowerLaw  
**File:** tfamily star_formation.IMF.piecewise_power_law.F90  
**Default value:** -2.35  
**Description:** The exponents used to define a piecewise power-law initial mass function.

**Name:** tfamily imfPiecewisePowerLawMassPoints  
**Attached to:** tfamily module:Star_Formation_IMF_PiecewisePowerLaw  
**File:** tfamily star_formation.IMF.piecewise_power_law.F90
**Default value:** 0.1, 125  
**Description:** The mass points used to define a piecewise power-law initial mass function.

**Name:** tfamily imfPiecewisePowerLawRecycledInstantaneous  
**Attached to:** tfamily module:Star_Formation_IMF_PiecewisePowerLaw  
**File:** tfamily star_formation.IMF.piecewise_power_law.F90  
**Default value:** 0.39  
**Description:** The recycled fraction for piecewise power-law stellar initial mass functions in the instantaneous recycling approximation.

**Name:** tfamily imfPiecewisePowerLawYieldInstantaneous  
**Attached to:** tfamily module:Star_Formation_IMF_PiecewisePowerLaw  
**File:** tfamily star_formation.IMF.piecewise_power_law.F90  
**Default value:** 0.02  
**Description:** The yield for piecewise power-law stellar initial mass functions in the instantaneous recycling approximation.

**Name:** tfamily imfSalpeterRecycledInstantaneous  
**Attached to:** tfamily module:Star_Formation_IMF_Salpeter  
**File:** tfamily star_formation.IMF.Salpeter.F90  
**Default value:** 0.39  
**Description:** The recycled fraction for the Salpeter IMF in the instantaneous recycling approximation.

**Name:** tfamily imfSalpeterYieldInstantaneous  
**Attached to:** tfamily module:Star_Formation_IMF_Salpeter  
**File:** tfamily star_formation.IMF.Salpeter.F90  
**Default value:** 0.02  
**Description:** The yield for the Salpeter IMF in the instantaneous recycling approximation.

**Name:** tfamily imfScaloRecycledInstantaneous  
**Attached to:** tfamily module:Star_Formation_IMF_Scalo  
**File:** tfamily star_formation.IMF.Scalo.F90  
**Default value:** 0.24 (computed internally)  
**Description:** The recycled fraction for the Scalo IMF in the instantaneous recycling approximation.

**Name:** tfamily imfScaloYieldInstantaneous  
**Attached to:** tfamily module:Star_Formation_IMF_Scalo  
**File:** tfamily star_formation.IMF.Scalo.F90  
**Default value:** 0.086 (internally computed)  
**Description:** The yield for the Scalo IMF in the instantaneous recycling approximation.

**Name:** tfamily imfSelectionDisk  
**Attached to:** tfamily module:Star_Formation_IMF_Select_Disk_Spheroid  
**File:** tfamily star_formation.IMF.select.diskSpheroid.F90  
**Default value:** Salpeter  
**Description:** The name of the initial mass function to use in the “diskSpheroid initial mass function” module for star formation in disks.

**Name:** tfamily imfSelectionFixed
4. Input Parameters

**Attached to:** tfamily module:Star_Formation_IMF_Select_Fixed  
**File:** tfamily star_formation.IMF.select.fixed.F90  
**Default value:** Chabrier  
**Description:** The name of the initial mass function to use in the “fixed initial mass function” module.

**Name:** tfamily imfSelectionMethod  
**Attached to:** tfamily module:Star_Formation_IMF  
**File:** tfamily star_formation.IMF.F90  
**Default value:** fixed  
**Description:** The name of the method to be used for selecting which IMF to use.

**Name:** tfamily imfSelectionSpheroid  
**Attached to:** tfamily module:Star_Formation_IMF_Select_Disk_Spheroid  
**File:** tfamily star_formation.IMF.select.diskSpheroid.F90  
**Default value:** Salpeter  
**Description:** The name of the initial mass function to use in the “diskSpheroid initial mass function” module for star formation in spheroids.

**Name:** tfamily infallRadiusMethod  
**Attached to:** tfamily module:Cooling_Infall_Radii  
**File:** tfamily cooling.infall_radius.F90  
**Default value:** coolingRadius  
**Description:** The name of the method to be used for calculations of the infall radius for cooling calculations.

**Name:** tfamily initialMassForSupernovaeTypeII  
**Attached to:** tfamily module:Stellar_Feedback_Standard  
**File:** tfamily stellar.astrophysics.feedback.standard.F90  
**Default value:** 8  
**Description:** The minimum mass that a star must have in order that it result in a Type II supernova.

**Name:** tfamily intergalacticMediumStateMethod  
**Attached to:** tfamily module:Intergalactic_Medium_State  
**File:** tfamily intergalactic_medium.state.F90  
**Default value:** recFast  
**Description:** The method to be used for intergalacticMediumState.

**Name:** tfamily intergalacticMediumStateFileName  
**Attached to:** tfamily module:Intergalactic_Medium_State  
**File:** tfamily intergalactic_medium.state.F90  
**Default value:** none  
**Description:** The name of the file from which to read intergalactic medium state data.

**Name:** tfamily linearGrowthMethod  
**Attached to:** tfamily module:Linear_Growth  
**File:** tfamily structure_formation.linear_growth.F90  
**Default value:** simple  
**Description:** The name of the method to be used for calculations of the linear growth factor.
Name: tfamily lognormalSpinDistributionMedian
Attached to: tfamily module:Halo_Spin_Distributions_Lognormal
File: tfamily dark_matter_halos.spins.distributions.lognormal.F90
Default value: 0.03687 [Bett et al., 2007]
Description: The median in a lognormal halo spin distribution.

Name: tfamily lognormalSpinDistributionSigma
Attached to: tfamily module:Halo_Spin_Distributions_Lognormal
File: tfamily dark_matter_halos.spins.distributions.lognormal.F90
Default value: 0.2216 [Bett et al., 2007]
Description: The dispersion in a lognormal halo spin distribution.

Name: tfamily luminosityBandRedshift
Attached to: tfamily module:Stellar_Luminosities_Structure
File: tfamily objects.stellar_luminosities.F90
Default value: none
Description: If present, force filters to be shifted to this redshift rather than that specified by [luminosityRedshift]. Allows sampling of the SED at wavelengths corresponding to other redshifts.

Name: tfamily luminosityFilter
Attached to: tfamily module:Stellar_Luminosities_Structure
File: tfamily objects.stellar_luminosities.F90
Default value: none
Description: The filter name for each stellar luminosity to be computed.

Name: tfamily luminosityFilterAbsoluteMagnitudeThresholdMaxima
Attached to: tfamily module:Galacticus_Merger_Tree_Output_Filter_Luminosities
File: tfamily galacticus.output.merger_tree.filters.luminosities.F90
Default value: none
Description: The maximum absolute magnitudes (in the AB system) of a galaxy to pass the luminosity output filter.

Name: tfamily luminosityFilterAbsoluteMagnitudeThresholdMinima
Attached to: tfamily module:Galacticus_Merger_Tree_Output_Filter_Luminosities
File: tfamily galacticus.output.merger_tree.filters.luminosities.F90
Default value: none
Description: The minimum absolute magnitudes (in the AB system) of a galaxy to pass the luminosity output filter.

Name: tfamily luminosityOutputOption
Attached to: tfamily module:Stellar_Luminosities_Structure
File: tfamily objects.stellar_luminosities.F90
Default value: present
Description: Selects which luminosities will be output at each output time:

all  Output all luminosities;

future  Output only those luminosities computed for the present output or future times;

present  Output only those luminosities computed for the present output time.
4. Input Parameters

Name: tfamily luminosityPostprocessSet
Attached to: tfamily module:Stellar_Luminosities_Structure
File: tfamily objects.stellar_luminosities.F90
Default value: none
Description: The name of the set of postprocessing algorithms to apply to this filter.

Name: tfamily luminosityRedshift
Attached to: tfamily module:Stellar_Luminosities_Structure
File: tfamily objects.stellar_luminosities.F90
Default value: none
Description: The redshift for which to compute each specified stellar luminosity.

Name: tfamily luminosityType
Attached to: tfamily module:Stellar_Luminosities_Structure
File: tfamily objects.stellar_luminosities.F90
Default value: none
Description: The luminosity type for each stellar luminosity to be computed:

rest  Compute luminosity in the galaxy rest frame;

observed  Compute luminosity in the observer frame.

Name: tfamily majorMergerMassRatio
Attached to: tfamily module:Satellite_Merging_Mass_Movements_Simple
File: tfamily satellites.merging.mass_movements.simple.F90
Default value: 0.25
Description: The mass ratio above which mergers are considered to be “major” in the simple merger mass movements method.

Name: tfamily massAccretionHistoryOutput
Attached to: tfamily module:Merger_Tree_Mass_Accretion_History
File: tfamily merger_trees.mass_accretion_history.F90
Default value: false
Description: Specifies whether or not to output mass accretion histories for the main branches of merger trees.

Name: tfamily massFlowStatisticsResetOnOutput
Attached to: tfamily module:Node_Component_Mass_Flow_Statistics_Standard
File: tfamily objects.nodes.components.mass_flow_statistics.standard.F90
Default value: true
Description: Specifies whether or not mass flow statistics should be reset to zero at each output.

Name: tfamily massFunctionCovarianceBinCount
Attached to: tfamily program:Mass_Function_Covariance
File: tfamily Mass_Function_Covariance.F90
Default value: 0.0
Description: The number of bins in the mass function for covariance calculations.

---

The luminosity computed in this way is that in the galaxy rest frame using a filter blueshifted to the galaxy’s redshift. This means that to compute an apparent magnitude you must add not only the distance modulus, but a factor of $-2.5 \log_{10}(1 + z)$ to account for compression of photon frequencies.
Name: tfamily massFunctionCovarianceFFTGridSize
Attached to: tfamily module:Statistics_Mass_Function_Covariance
File: tfamily statistics.mass_function.covariance.F90
Default value: 64
Description: The size of the FFT grid to use in computing window functions for mass function covariance matrices.

Name: tfamily massFunctionCovarianceHaloMassMaximum
Attached to: tfamily module:Statistics_Mass_Function_Covariance
File: tfamily statistics.mass_function.covariance.F90
Default value: $10^{10}\,M_\odot$
Description: The minimum halo mass to use when computing mass function covariance matrices.

Name: tfamily massFunctionCovarianceHaloMassMinimum
Attached to: tfamily module:Statistics_Mass_Function_Covariance
File: tfamily statistics.mass_function.covariance.F90
Default value: $10^{10}\,M_\odot$
Description: The minimum halo mass to use when computing mass function covariance matrices.

Name: tfamily massFunctionCovarianceIncludeHalo
Attached to: tfamily program:Mass_Function_Covariance
File: tfamily Mass_Function_Covariance.F90
Default value: true
Description: Specifies whether or not to include the halo contribution to mass function covariance matrices.

Name: tfamily massFunctionCovarianceIncludeLSS
Attached to: tfamily program:Mass_Function_Covariance
File: tfamily Mass_Function_Covariance.F90
Default value: true
Description: Specifies whether or not to include the large-scale structure contribution to mass function covariance matrices.

Name: tfamily massFunctionCovarianceIncludePoisson
Attached to: tfamily program:Mass_Function_Covariance
File: tfamily Mass_Function_Covariance.F90
Default value: true
Description: Specifies whether or not to include the Poisson contribution to mass function covariance matrices.

Name: tfamily massFunctionCovarianceMassMaximum
Attached to: tfamily program:Mass_Function_Covariance
File: tfamily Mass_Function_Covariance.F90
Default value: $10^8\,M_\odot$
Description: The maximum mass in the mass function for covariance calculations.

Name: tfamily massFunctionCovarianceMassMinimum
Attached to: tfamily program:Mass_Function_Covariance
File: tfamily Mass_Function_Covariance.F90
4. Input Parameters

Default value: \(10^8 M_\odot\)
Description: The minimum mass in the mass function for covariance calculations.

Name: tfamily massFunctionCovarianceOutputFileName
Attached to: tfamily program:Mass_Function_Covariance
File: tfamily Mass_Function_Covariance.F90
Default value: covariance.hdf5
Description: The name of the file to which the covariance matrix should be written.

Name: tfamily massFunctionCovarianceRedshiftMaximum
Attached to: tfamily program:Mass_Function_Covariance
File: tfamily Mass_Function_Covariance.F90
Default value: 0.0
Description: The maximum redshift at which calculations of the mass function covariance should be carried out.

Name: tfamily massFunctionCovarianceRedshiftMinimum
Attached to: tfamily program:Mass_Function_Covariance
File: tfamily Mass_Function_Covariance.F90
Default value: 0.0
Description: The minimum redshift at which calculations of the mass function covariance should be carried out.

Name: tfamily massVarianceFilteredPowerSpectrumTolerance
Attached to: tfamily module:Cosmological_Mass_Variance_Filtered_Power_Spectrum
File: tfamily structure_formation.cosmological_mass_variance.filtered_power_spectrum.F90
Default value: \(4 \times 10^{-6}\)
Description: The relative tolerance to use in integrating over the linear power spectrum to compute the cosmological mass variance.

Name: tfamily massVarianceFilteredPowerSpectrumTopHatTolerance
Attached to: tfamily module:Cosmological_Mass_Variance_Filtered_Power_Spectrum
File: tfamily structure_formation.cosmological_mass_variance.filtered_power_spectrum.F90
Default value: \(10^{-6}\)
Description: The relative tolerance to use in integrating over the linear power spectrum using a top-hat (real space) window function to compute the cosmological mass variance.

Name: tfamily mergeTargetTimeOffsetMaximumAbsolute
Attached to: tfamily module:Merger_Tree_Timesteps_Satellite
File: tfamily merger_trees.evolve.timesteps.satellite.F90
Default value: 0.01
Description: The maximum absolute time difference (in Gyr) allowed between merging pairs of galaxies.

Name: tfamily mergeTargetTimeOffsetMaximumRelative
Attached to: tfamily module:Merger_Tree_Timesteps_Satellite
File: tfamily merger_trees.evolve.timesteps.satellite.F90
Default value: 0.001
Description: The maximum time difference (relative to the cosmic time at the merger epoch) allowed between merging pairs of galaxies.
Name: tfamily mergerRemnantRadiativeEfficiency
Attached to: tfamily module: Satellite_Merging_Remnant_Sizes_Covington2008
Default value: 2.75 [Covington et al., 2008]
Description: The coefficient, $C_{rad}$ energy used in the Covington et al. [2008] merger remnant size algorithm.

Name: tfamily mergerRemnantSizeOrbitalEnergy
Attached to: tfamily module: Satellite_Merging_Remnant_Sizes_Cole2000
Default value: 1
Description: The orbital energy used in the “Cole2000” merger remnant sizes calculation in units of the characteristic orbital energy.

Name: tfamily mergerTreeAnalyses
Attached to: tfamily module: Galacticus_Output_Merger_Tree
File: tfamily galacticus.output.merger_tree.F90
Default value: none
Description: List of analyses to carry out on merger trees.

Name: tfamily mergerTreeBaseRedshift
Attached to: tfamily module: Merger_Tree_Smooth_Accretion
File: tfamily merger_trees.construct.smooth_accretion.F90
Default value: 0
Description: The redshift at which to plant the base node when building the smoothly accreting merger tree.

Name: tfamily mergerTreeBuildCole2000AccretionLimit
Attached to: tfamily module: Merger_Tree_Build_Cole2000
Default value: 0.1
Description: The largest fractional mass change due to subresolution accretion allowed in a timestep in merger trees built by the Cole et al. [2000] method.

Name: tfamily mergerTreeBuildCole2000FixedRandomSeeds
Attached to: tfamily module: Merger_Tree_Build_Cole2000
Default value: false
Description: Specifies whether the random number sequence should be restarted for each tree using a deterministically derived (from the tree index) seed. This allows the exact same tree to be generated even when running multiple threads.

Name: tfamily mergerTreeBuildCole2000HighestRedshift
Attached to: tfamily module: Merger_Tree_Build_Cole2000
Default value: 10^5
Description: The highest redshift to which merger trees will be built in the Cole et al. [2000] method.
4. Input Parameters

Name: tfamily mergerTreeBuildCole2000MergeProbability
Attached to: tfamily module:Merger_Tree_Build_Cole2000
Default value: 0.1
Description: The largest probability of branching allowed in a timestep in merger trees built by the Cole et al. [2000] method.

Name: tfamily mergerTreeBuildHaloMassMaximum
Attached to: tfamily module:Merger_Tree_Build
File: tfamily merger_trees.construct.build.F90
Default value: $10^{15}$
Description: The maximum mass of merger tree base halos to consider when building merger trees, in units of $M_\odot$.

Name: tfamily mergerTreeBuildHaloMassMinimum
Attached to: tfamily module:Merger_Tree_Build
File: tfamily merger_trees.construct.build.F90
Default value: $10^{10}$
Description: The minimum mass of merger tree base halos to consider when building merger trees, in units of $M_\odot$.

Name: tfamily mergerTreeBuildMassResolutionFixed
Attached to: tfamily module:Merger_Trees_Build_Mass_Resolution_Fixed
File: tfamily merger_trees.construct.build.mass_resolution.fixed.F90
Default value: $5 \times 10^9$
Description: The minimum mass (in units of $M_\odot$) of halos to be resolved in merger trees that are built.

Name: tfamily mergerTreeBuildMassResolutionScaledFraction
Attached to: tfamily module:Merger_Trees_Build_Mass_Resolution_Scaled
File: tfamily merger_trees.construct.build.mass_resolution.scaled.F90
Default value: $10^{-3}$
Description: The fraction of the tree’s root node mass to be used for the mass resolution in merger trees that are built using the “scaled” mass resolution algorithm.

Name: tfamily mergerTreeBuildMassResolutionScaledMinimum
Attached to: tfamily module:Merger_Trees_Build_Mass_Resolution_Scaled
File: tfamily merger_trees.construct.build.mass_resolution.scaled.F90
Default value: $5 \times 10^9$
Description: The minimum mass (in units of $M_\odot$) of halos to be resolved in merger trees that are built using the “scaled” mass resolution algorithm.

Name: tfamily mergerTreeBuildMethod
Attached to: tfamily module:Merger_Tree_Build
File: tfamily merger_trees.construct.build.F90
Default value: Cole2000
Description: The name of the method to be used to build merger trees.

Name: tfamily mergerTreeBuildTreeMassesFile
Attached to: tfamily subroutine:Merger_Tree_Build_Initialize
File: tfamily merger_trees.construct.build.F90  
Default value: null  
Description: Specifies the name of a file from which to read the masses of merger tree root halos when building merger trees.

Name: tfamily mergerTreeBuildTreesBaseRedshift  
Attached to: tfamily module:Merger_Tree_Build  
File: tfamily merger_trees.construct.build.F90  
Default value: 0  
Description: The redshift at which to plant the base node when building merger trees.

Name: tfamily mergerTreeBuildTreesBeginAtTree  
Attached to: tfamily module:Merger_Tree_Build  
File: tfamily merger_trees.construct.build.F90  
Default value: 1 (if processing trees in ascending order), or equal to the number of trees (otherwise)  
Description: The index (in order of increasing base halo mass) of the tree at which to begin when building merger trees.

Name: tfamily mergerTreeBuildTreesHaloMassDistribution  
Attached to: tfamily module:Merger_Tree_Build  
File: tfamily merger_trees.construct.build.F90  
Default value: uniform  
Description: The method to be used to construct a distribution of base halo masses.

Name: tfamily mergerTreeBuildTreesHaloMassExponent  
Attached to: tfamily module:Merger_Trees_Mass_Function_Sampling_Power_Law  
File: tfamily merger_trees.construct.mass_function_sampling.power_law.F90  
Default value: 1  
Description: Halo masses will be (pseudo-)uniformly distributed in $[\log(M)]^{1/(1+\alpha)}$ where $\alpha =$mergerTreeBuildTreesHaloMassExponent.

Name: tfamily mergerTreeBuildTreesHaloMassGaussianMean  
Attached to: tfamily module:Merger_Trees_Mass_Function_Sampling_Gaussian  
File: tfamily merger_trees.construct.mass_function_sampling.gaussian.F90  
Default value: none  
Description: The mean mass of halo to simulate when using a Gaussian sampling of the halo mass function.

Name: tfamily mergerTreeBuildTreesHaloMassGaussianSigma  
Attached to: tfamily module:Merger_Trees_Mass_Function_Sampling_Gaussian  
File: tfamily merger_trees.construct.mass_function_sampling.gaussian.F90  
Default value: none  
Description: The dispersion in mass of halo to simulate when using a Gaussian sampling of the halo mass function.

Name: tfamily mergerTreeBuildTreesPerDecade  
Attached to: tfamily module:Merger_Tree_Build  
File: tfamily merger_trees.construct.build.F90  
Default value: 10  
Description: The number of merger trees to build per decade of base halo mass.
### 4. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Associated with</th>
<th>File</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tfamily mergerTreeBuildTreesProcessDescending</code></td>
<td><code>tfamily module:Merger_Tree_Build</code></td>
<td><code>tfamily merger_trees.construct.build.F90</code></td>
<td><code>true</code></td>
<td>If true, causes merger trees to be processed in order of decreasing mass.</td>
</tr>
<tr>
<td><code>tfamily mergerTreeComputeConditionalMassFunction</code></td>
<td><code>tfamily module:Merger_Trees_Conditional_Mass_Function</code></td>
<td><code>tfamily merger_trees.conditional_mass_functions.F90</code></td>
<td><code>false</code></td>
<td>Specifies whether or not the conditional mass function of merger trees should be computed.</td>
</tr>
<tr>
<td><code>tfamily mergerTreeComputeConditionalMassFunctionMassRatioCount</code></td>
<td><code>tfamily module:Merger_Trees_Conditional_Mass_Function</code></td>
<td><code>tfamily merger_trees.conditional_mass_functions.F90</code></td>
<td><code>10</code></td>
<td>The number of bins in mass ratio when constructing conditional halo mass functions.</td>
</tr>
<tr>
<td><code>tfamily mergerTreeComputeConditionalMassFunctionMassRatioMaximum</code></td>
<td><code>tfamily module:Merger_Trees_Conditional_Mass_Function</code></td>
<td><code>tfamily merger_trees.conditional_mass_functions.F90</code></td>
<td>$10^{15}M_\odot$</td>
<td>The maximum mass ratio to bin when constructing conditional halo mass functions.</td>
</tr>
<tr>
<td><code>tfamily mergerTreeComputeConditionalMassFunctionMassRatioMinimum</code></td>
<td><code>tfamily module:Merger_Trees_Conditional_Mass_Function</code></td>
<td><code>tfamily merger_trees.conditional_mass_functions.F90</code></td>
<td>$10^{10}M_\odot$</td>
<td>The minimum mass ratio to bin when constructing conditional halo mass functions.</td>
</tr>
<tr>
<td><code>tfamily mergerTreeComputeConditionalMassFunctionParentMassCount</code></td>
<td><code>tfamily module:Merger_Trees_Conditional_Mass_Function</code></td>
<td><code>tfamily merger_trees.conditional_mass_functions.F90</code></td>
<td><code>10</code></td>
<td>The number of bins in parent mass when constructing conditional halo mass functions.</td>
</tr>
<tr>
<td><code>tfamily mergerTreeComputeConditionalMassFunctionParentMassMaximum</code></td>
<td><code>tfamily module:Merger_Trees_Conditional_Mass_Function</code></td>
<td><code>tfamily merger_trees.conditional_mass_functions.F90</code></td>
<td>$10^{15}M_\odot$</td>
<td>The maximum parent halo mass to bin when constructing conditional halo mass functions.</td>
</tr>
<tr>
<td><code>tfamily mergerTreeComputeConditionalMassFunctionParentMassMinimum</code></td>
<td><code>tfamily module:Merger_Trees_Conditional_Mass_Function</code></td>
<td><code>tfamily merger_trees.conditional_mass_functions.F90</code></td>
<td>$10^{10}M_\odot$</td>
<td>The minimum parent halo mass to bin when constructing conditional halo mass functions.</td>
</tr>
</tbody>
</table>
Name: tfamily mergerTreeComputeConditionalMassFunctionParentRedshifts
Attached to: tfamily module:Merger_Trees_Conditional_Mass_Function
File: tfamily merger_trees.conditional_mass_functions.F90
Default value: 0.0
Description: The set of parent halo redshifts to use when constructing conditional halo mass functions.

Name: tfamily mergerTreeComputeConditionalMassFunctionPrimaryProgenitorDepth
Attached to: tfamily module:Merger_Trees_Conditional_Mass_Function
File: tfamily merger_trees.conditional_mass_functions.F90
Default value: 2
Description: The depth in progenitor ranking for which to store ranked progenitor mass functions. For example, a value of 2 means store mass functions for the most massive, and second most massive progenitor.

Name: tfamily mergerTreeComputeConditionalMassFunctionProgenitorRedshifts
Attached to: tfamily module:Merger_Trees_Conditional_Mass_Function
File: tfamily merger_trees.conditional_mass_functions.F90
Default value: 1.0
Description: The set of progenitor halo redshifts to use when constructing conditional halo mass functions.

Name: tfamily mergerTreeConditionalMassFunctionFormationRateTimeFraction
Attached to: tfamily module:Merger_Trees_Conditional_Mass_Function
File: tfamily merger_trees.conditional_mass_functions.F90
Default value: 0.01
Description: The fraction of the current time over which to estimate the formation rate of halos when computing merger tree statistics.

Name: tfamily mergerTreeConstructFullySpecifiedFileName
Attached to: tfamily module:Merger_Trees_Construct_Fully_Specified
File: tfamily merger_trees.construct.fully_specified.F90
Default value: none
Description: The name of the file giving the fully-specified description of the merger tree to process.

Name: tfamily mergerTreeConstructMethod
Attached to: tfamily module:Merger_Tree_Construction
File: tfamily merger_trees.construct.F90
Default value: build
Description: Selects the method to be used for constructing merger trees.

Name: tfamily mergerTreeEnforceMonotonicGrowth
Attached to: tfamily module:Merger_Trees_Monotonic_Mass_Growth
File: tfamily merger_trees.monotonic_mass_growth.F90
Default value: false
Description: Specifies whether or not to enforce monotonic mass growth along the branches of merger trees.

Name: tfamily mergerTreeEvolutionDump
Attached to: tfamily module:Merger_Trees_Dump_Evolution
File: tfamily merger_trees.dump_evolution.F90
4. Input Parameters

Default value: false
Description: Specifies whether or not to output the evolution of merger trees.

Name: tfamily mergerTreeEvolutionDumpFileName
Attached to: tfamily module:Merger_Trees_Dump_Evolution
File: tfamily merger_trees.dump_evolution.F90
Default value: mergerTreeEvolution.xml
Description: Specifies the file to which merger tree evolution should be dumped.

Name: tfamily mergerTreeExportFileName
Attached to: tfamily module:Merger_Trees_Write
File: tfamily merger_trees.write.F90
Default value: galacticusExportedTrees.hdf5
Description: The name of the file to which merger trees will be exported.

Name: tfamily mergerTreeExportOutputFormat
Attached to: tfamily module:Merger_Trees_Write
File: tfamily merger_trees.write.F90
Default value: galacticus
Description: The output format to use when exporting merger trees.

Name: tfamily mergerTreeHaloMass
Attached to: tfamily module:Merger_Tree_Smooth_Accretion
File: tfamily merger_trees.construct.smooth_accretion.F90
Default value: $10^{12}$
Description: The final mass of the merger tree base halo to consider when building a smoothly accreting merger tree, in units of $M_\odot$.

Name: tfamily mergerTreeHaloMassDeclineFactor
Attached to: tfamily module:Merger_Tree_Smooth_Accretion
File: tfamily merger_trees.construct.smooth_accretion.F90
Default value: 0.9
Description: The factor by which halo mass should decrease in each step back in time building a smoothly accreting merger tree, in units of $M_\odot$.

Name: tfamily mergerTreeHaloMassResolution
Attached to: tfamily module:Merger_Tree_Smooth_Accretion
File: tfamily merger_trees.construct.smooth_accretion.F90
Default value: $10^{12}$
Description: The final mass of the merger tree base halo to consider when building a smoothly accreting merger tree, in units of $M_\odot$.

Name: tfamily mergerTreeImportGalacticusMismatchIsFatal
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: true
Description: Specifies whether mismatches in cosmological parameter values between GALACTICUS and the merger tree file should be considered fatal.
Name: tfamily mergerTreeImportGalacticusReweightTrees
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: false
Description: Specifies whether merger tree weights should be recomputed from the halo mass function.

Name: tfamily mergerTreeImportSussingBadValue
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: -0.5
Description: Use for bad value detection in “Sussing” merger trees. Values for scale radius and halo spin which exceed this threshold are assumed to be bad.

Name: tfamily mergerTreeImportSussingBadValueTest
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: lessThan
Description: Use for bad value detection in “Sussing” merger trees. Values which exceed the threshold in the specified direction are assumed to be bad.

Name: tfamily mergerTreeImportSussingBinaryFormatOld
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: true
Description: Specifies whether the old binary format is to be used (for reading only).

Name: tfamily mergerTreeImportSussingConvertToBinary
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: true
Description: Specifies whether halo and tree files in the “Sussing” format should be converted to binary the first time they are read and stored to file. This allows rapid re-reading in future.

Name: tfamily mergerTreeImportSussingForestFile
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: none
Description: Name of file containing data on number of halos in each forest.

Name: tfamily mergerTreeImportSussingForestFirst
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: 1
Description: Index of first forest to include.

Name: tfamily mergerTreeImportSussingForestLast
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: -1
4. Input Parameters

Description: Index of last forest to include.

Name: tfamily mergerTreeImportSussingForestReverseSnapshotOrder
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: false
Description: If true, the order of forest snapshots will be reversed after being read. This may be necessary to cause them to match the order of snapshot files.

Name: tfamily mergerTreeImportSussingMassOption
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: default
Description: Mass option for Sussing merger trees.

Name: tfamily mergerTreeImportSussingMismatchIsFatal
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: true
Description: Specifies whether mismatches in cosmological parameter values between GALACTICUS and “Sussing Merger Trees” format [Srisawat et al., 2013] merger tree files should be considered fatal.

Name: tfamily mergerTreeImportSussingNonTreeNodeIsFatal
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: true
Description: Specifies whether nodes in snapshot files but not in the merger tree file should be considered fatal when importing from the “Sussing Merger Trees” format [Srisawat et al., 2013].

Name: tfamily mergerTreeImportSussingSubvolumeBuffer
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: 0.0
Description: Specifies the buffer region (in units of Mpc/h to follow the format convention) around subvolumes of a “Sussing Merger Trees” format [Srisawat et al., 2013] merger tree file which should be read in to ensure that no halos are missed from trees.

Name: tfamily mergerTreeImportSussingSubvolumeCount
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: 1
Description: Specifies the number of subvolumes along each axis into which a “Sussing Merger Trees” format [Srisawat et al., 2013] merger tree files should be split for processing through GALACTICUS.

Name: tfamily mergerTreeImportSussingSubvolumeIndex
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: [0,0,0]
Description: Specifies the index (in each dimension) of the subvolume of a “Sussing Merger Trees” format
[Srisawat et al., 2013] merger tree file to process. Indices range from 0 to [mergerTreeImportSussingSubvolumeCount]−1.

Name: tfamily mergerTreeImporterMethod
Attached to: tfamily module:Merger_Tree_Read_Importers
File: tfamily merger_trees.construct.read.importer.F90
Default value: galacticus
Description: The method to be used for mergerTreeImporter.

Name: tfamily mergerTreeOutput
Attached to: tfamily module:Galacticus_Output_Merger_Tree
File: tfamily galacticus.output.merger_tree.F90
Default value: true
Description: Specifies whether or not to output galaxies.

Name: tfamily mergerTreeOutputFilters
Attached to: tfamily module:Galacticus_Merger_Tree_Output_Filters
File: tfamily galacticus.output.merger_tree.filters.F90
Default value: none
Description: A list of filters that should be applied when deciding which galaxies to output.

Name: tfamily mergerTreeOutputReferences
Attached to: tfamily module:Galacticus_Output_Merger_Tree
File: tfamily galacticus.output.merger_tree.F90
Default value: false
Description: Specifies whether or not references to individual merger tree datasets should be output.

Name: tfamily mergerTreePruneBranches
Attached to: tfamily module:Merger_Trees_Prune_Branches
File: tfamily merger_trees.prune_branches.F90
Default value: false
Description: Specifies whether or not to prune merger trees prior to evolution.

Name: tfamily mergerTreePruneHierarchyAtDepth
Attached to: tfamily module:Merger_Trees_Prune_Hierarchy
File: tfamily merger_trees.prune_hierarchy.F90
Default value: 0
Description: The depth in the hierarchy at which to prune merger trees. (Zero indicates to not prune.)

Name: tfamily mergerTreePruneNonEssential
Attached to: tfamily module:Merger_Trees_Prune_Non_Essential
File: tfamily merger_trees.prune_non_essential.F90
Default value: false
Description: Specifies whether or not to prune non-essential merger trees branches prior to evolution.

Name: tfamily mergerTreePruningMassThreshold
Attached to: tfamily module:Merger_Trees_Prune_Branches
File: tfamily merger_trees.prune_branches.F90
Default value: 0
Description: Threshold mass below which merger tree branches should be pruned.
4. Input Parameters

Name: tfamily mergerTreePruningNonEssentialID
Attached to: tfamily module: Merger_Trees_Prune_Non_Essential
File: tfamily merger_trees.prune_non_essential.F90
Default value: none
Description: ID of the essential node to avoid pruning.

Name: tfamily mergerTreePruningNonEssentialTime
Attached to: tfamily module: Merger_Trees_Prune_Non_Essential
File: tfamily merger_trees.prune_non_essential.F90
Default value: none
Description: Time of the essential node to avoid pruning.

Name: tfamily mergerTreeReadAllowBranchJumps
Attached to: tfamily module: Merger_Tree_Read
File: tfamily merger_trees.construct.read.F90
Default value: true
Description: Specifies whether nodes are allowed to jump between branches.

Name: tfamily mergerTreeReadAllowSubhaloPromotions
Attached to: tfamily module: Merger_Tree_Read
File: tfamily merger_trees.construct.read.F90
Default value: true
Description: Specifies whether subhalos are permitted to be promoted to being isolated halos.

Name: tfamily mergerTreeReadBeginAt
Attached to: tfamily module: Merger_Tree_Read
File: tfamily merger_trees.construct.read.F90
Default value: -1
Description: Specifies the index of the tree to begin at. (Use -1 to always begin with the first tree.)

Name: tfamily mergerTreeReadConcentrationFallbackMethod
Attached to: tfamily module: Merger_Tree_Read
File: tfamily merger_trees.construct.read.F90
Default value: [darkMatterProfileConcentrationMethod]
Description: The method to be used for setting node scale radii when reading merger trees from file and the node mass falls below the reliability threshold.

Name: tfamily mergerTreeReadFileName
Attached to: tfamily module: Merger_Tree_Read
File: tfamily merger_trees.construct.read.F90
Default value: none
Description: The name of the file from which merger tree data should be read when using the
[mergerTreeConstructMethod]=read tree construction method.

Name: tfamily mergerTreeReadMissingHostsAreFatal
Attached to: tfamily module: Merger_Tree_Read
File: tfamily merger_trees.construct.read.F90
Default value: true
Description: Specifies whether nodes with missing host nodes should be considered to be fatal—see §6.2.1.

Name: tfamily mergerTreeReadOutputTimeSnapTolerance
Attached to: tfamily module: Merger_Tree_Read
File: tfamily merger_trees.construct.read.F90
Default value: 0
Description: The relative tolerance required to “snap” a node time to the closest output time.

Name: tfamily mergerTreeReadPresetMergerNodes
Attached to: tfamily module: Merger_Tree_Read
File: tfamily merger_trees.construct.read.F90
Default value: true
Description: Specifies whether the target nodes for mergers should be preset (i.e. determined from descendent nodes). If they are not, merging will be with each satellite’s host node.

Name: tfamily mergerTreeReadPresetMergerTimes
Attached to: tfamily module: Merger_Tree_Read
File: tfamily merger_trees.construct.read.F90
Default value: true
Description: Specifies whether merging times for subhalos should be preset when reading merger trees from a file.

Name: tfamily mergerTreeReadPresetOrbits
Attached to: tfamily module: Merger_Tree_Read
File: tfamily merger_trees.construct.read.F90
Default value: true
Description: Specifies whether node orbits should be preset when reading merger trees from a file.

Name: tfamily mergerTreeReadPresetOrbitsAssertAllSet
Attached to: tfamily module: Merger_Tree_Read
File: tfamily merger_trees.construct.read.F90
Default value: true
Description: Asserts that all virial orbits must be preset. If any can not be set, GALACTICUS will stop.

Name: tfamily mergerTreeReadPresetOrbitsBoundOnly
Attached to: tfamily module: Merger_Tree_Read
File: tfamily merger_trees.construct.read.F90
Default value: true
Description: Specifies whether only bound node orbits should be set.

Name: tfamily mergerTreeReadPresetOrbitsSetAll
Attached to: tfamily module: Merger_Tree_Read
File: tfamily merger_trees.construct.read.F90
Default value: true
Description: Forces all orbits to be set. If the computed orbit does not cross the virial radius, then select one at random instead.

Name: tfamily mergerTreeReadPresetParticleCounts
4. Input Parameters

**Attached to:** tfamily module:Merger_Tree_Read  
**File:** tfamily merger_trees.construct.read.F90  
**Default value:** false  
**Description:** Specifies whether node particle counts should be preset when reading merger trees from a file.

**Name:** tfamily mergerTreeReadPresetPositions  
**Attached to:** tfamily module:Merger_Tree_Read  
**File:** tfamily merger_trees.construct.read.F90  
**Default value:** true  
**Description:** Specifies whether node positions should be preset when reading merger trees from a file.

**Name:** tfamily mergerTreeReadPresetScaleRadii  
**Attached to:** tfamily module:Merger_Tree_Read  
**File:** tfamily merger_trees.construct.read.F90  
**Default value:** true  
**Description:** Specifies whether node scale radii should be preset when reading merger trees from a file.

**Name:** tfamily mergerTreeReadPresetScaleRadiiConcentrationMaximum  
**Attached to:** tfamily module:Merger_Tree_Read  
**File:** tfamily merger_trees.construct.read.F90  
**Default value:** 60  
**Description:** The largest concentration ($c = r_{vir}/r_s$) allowed when setting scale radii, $r_s$.

**Name:** tfamily mergerTreeReadPresetScaleRadiiConcentrationMinimum  
**Attached to:** tfamily module:Merger_Tree_Read  
**File:** tfamily merger_trees.construct.read.F90  
**Default value:** 3  
**Description:** The lowest concentration ($c = r_{vir}/r_s$) allowed when setting scale radii, $r_s$.

**Name:** tfamily mergerTreeReadPresetScaleRadiiFailureIsFatal  
**Attached to:** tfamily module:Merger_Tree_Read  
**File:** tfamily merger_trees.construct.read.F90  
**Default value:** true  
**Description:** Specifies whether failure to set a node scale radii should be regarded as a fatal error. (If not, a fallback method to set scale radius is used in such cases.)

**Name:** tfamily mergerTreeReadPresetScaleRadiiMinimumMass  
**Attached to:** tfamily module:Merger_Tree_Read  
**File:** tfamily merger_trees.construct.read.F90  
**Default value:** 0  
**Description:** The minimum halo mass for which scale radii should be preset (if [mergerTreeReadPresetScaleRadii]=true).

**Name:** tfamily mergerTreeReadPresetSpins  
**Attached to:** tfamily module:Merger_Tree_Read  
**File:** tfamily merger_trees.construct.read.F90  
**Default value:** true  
**Description:** Specifies whether node spins should be preset when reading merger trees from a file.
Name: tfamily mergerTreeReadPresetSpins3D  
Attached to: tfamily module:Merger_Tree_Read  
File: tfamily merger_trees.construct.read.F90  
Default value: true  
Description: Specifies whether node 3-D spin vectors should be preset when reading merger trees from a file.

Name: tfamily mergerTreeReadPresetSubhaloIndices  
Attached to: tfamily module:Merger_Tree_Read  
File: tfamily merger_trees.construct.read.F90  
Default value: true  
Description: Specifies whether subhalo indices should be preset when reading merger trees from a file.

Name: tfamily mergerTreeReadPresetSubhaloMasses  
Attached to: tfamily module:Merger_Tree_Read  
File: tfamily merger_trees.construct.read.F90  
Default value: true  
Description: Specifies whether subhalo mass should be preset when reading merger trees from a file.

Name: tfamily mergerTreeReadPresetUnphysicalSpins  
Attached to: tfamily module:Merger_Tree_Read  
File: tfamily merger_trees.construct.read.F90  
Default value: false  
Description: When reading merger trees from file and presetting halo spins, detect unphysical (<=0) spins and preset them using the selected halo spin method.

Name: tfamily mergerTreeReadPresetVelocityDispersions  
Attached to: tfamily module:Merger_Tree_Read  
File: tfamily merger_trees.construct.read.F90  
Default value: false  
Description: Specifies whether node velocity dispersions should be preset when reading merger trees from a file.

Name: tfamily mergerTreeReadPresetVelocityMaxima  
Attached to: tfamily module:Merger_Tree_Read  
File: tfamily merger_trees.construct.read.F90  
Default value: false  
Description: Specifies whether node rotation curve velocity maxima should be preset when reading merger trees from a file.

Name: tfamily mergerTreeReadSubhaloAngularMomentaMethod  
Attached to: tfamily module:Merger_Tree_Read  
File: tfamily merger_trees.construct.read.F90  
Default value: true  
Description: Specifies whether subhalo mass should be preset when reading merger trees from a file.

Name: tfamily mergerTreeReadSubresolutionMergingMethod  
Attached to: tfamily module:Merger_Tree_Read  
File: tfamily merger_trees.construct.read.F90
4. Input Parameters

Default value: null
Description: The name of a satellite merging timescale method to be used for computing the extra time until merging for subhalos.

Name: tfamily mergerTreeReadTreeNodeToRootNodeIndex
Attached to: tfamily module: Merger_Tree_Read
File: tfamily merger_trees.construct.read.F90
Default value: false
Description: Specifies whether tree indices should always be set to the index of their root node.

Name: tfamily mergerTreeRegridCount
Attached to: tfamily module: Merger_Trees_Regrid_Times
File: tfamily merger_trees.regrid_times.F90
Default value: false
Description: Number of points in time to use when regridding merger trees.

Name: tfamily mergerTreeRegridDumpTrees
Attached to: tfamily module: Merger_Trees_Regrid_Times
File: tfamily merger_trees.regrid_times.F90
Default value: false
Description: Specifies whether or not to dump merger trees as they are regridded.

Name: tfamily mergerTreeRegridEndExpansionFactor
Attached to: tfamily module: Merger_Trees_Regrid_Times
File: tfamily merger_trees.regrid_times.F90
Default value: false
Description: Ending expansion factor to use when regridding merger trees.

Name: tfamily mergerTreeRegridRedshifts
Attached to: tfamily module: Merger_Trees_Regrid_Times
File: tfamily merger_trees.regrid_times.F90
Default value: none
Description: The redshifts at which merger trees are to be regridded when the `mergerTreeRegridSpacing` option is selected.

Name: tfamily mergerTreeRegridSpacing
Attached to: tfamily module: Merger_Trees_Regrid_Times
File: tfamily merger_trees.regrid_times.F90
Default value: false
Description: Type of spacing to use in merger tree regridding (linear or logarithmic).

Name: tfamily mergerTreeRegridStartExpansionFactor
Attached to: tfamily module: Merger_Trees_Regrid_Times
File: tfamily merger_trees.regrid_times.F90
Default value: false
Description: Starting expansion factor to use when regridding merger trees.

Name: tfamily mergerTreeRegridTimes
Attached to: tfamily module: Merger_Trees_Regrid_Times
File: tfamily merger_trees.regrid_times.F90
Default value: false
Description: Specifies whether or not to regrid merger tree times.

Name: tfamily mergerTreeStateStoreFile
Attached to: tfamily module:Merger_Trees_State_Store
File: tfamily merger_trees.construct.state_restore.F90
Default value: storedTree.dat
Description: The name of a file from which to restore a merger tree state.

Name: tfamily mergerTreeStructureDump
Attached to: tfamily module:Merger_Tree_Dump_Structure
File: tfamily merger_trees.dump_structure.F90
Default value: false
Description: Specifies whether or not to output the structure of merger trees prior to evolution.

Name: tfamily mergerTreeStructureDumpDirectory
Attached to: tfamily module:Merger_Tree_Dump_Structure
File: tfamily merger_trees.dump_structure.F90
Default value: .
Description: Specifies the directory to which merger tree structure should be dumped.

Name: tfamily mergerTreeStructureDumpMassMaximum
Attached to: tfamily module:Merger_Tree_Dump_Structure
File: tfamily merger_trees.dump_structure.F90
Default value: $10^{30} M_\odot$
Description: Specifies the minimum root mass for which merger tree structure should be dumped.

Name: tfamily mergerTreeStructureDumpMassMinimum
Attached to: tfamily module:Merger_Tree_Dump_Structure
File: tfamily merger_trees.dump_structure.F90
Default value: $0 M_\odot$
Description: Specifies the minimum root mass for which merger tree structure should be dumped.

Name: tfamily mergerTreeStructureOutput
Attached to: tfamily module:Merger_Tree_Output_Structure
File: tfamily merger_trees.output_structure.F90
Default value: false
Description: Specifies whether or not to output the structure of merger trees prior to evolution.

Name: tfamily mergerTreeStructureOutputDarkMatterProfileScale
Attached to: tfamily module:Node_Component_Dark_Matter_Profile_Scale
File: tfamily objects.nodes.components.dark_matter_profile.scale.F90
Default value: false
Description: Determines whether or not dark matter halo scale radius is included in outputs of merger trees.

Name: tfamily mergerTreeStructureOutputDarkMatterProfileShape
Attached to: tfamily module:Node_Component_Dark_Matter_Profile_Scale_Shape
4. Input Parameters

File: tfamily objects.nodes.components.dark_matter_profile.scale_shape.F90
Default value: false
Description: Determines whether or not dark matter halo shape parameter is included in outputs of merger trees.

Name: tfamily mergerTreeStructureOutputVirialQuantities
Attached to: tfamily module:Merger_Tree_Output_Structure
File: tfamily merger_trees.output_structure.F90
Default value: false
Description: Specifies whether or not to output virial quantities (radius and velocity) when outputting the structure of merger trees prior to evolution.

Name: tfamily mergerTreesBuildMassResolutionMethod
Attached to: tfamily module:Merger_Trees_Build_Mass_Resolution
File: tfamily merger_trees.construct.build.mass_resolution.F90
Default value: fixed
Description: The name of the method to be used for computing the mass resolution to use when building merger trees.

Name: tfamily mergerTreesDumpStructure
Attached to: tfamily module:Merger_Trees_Evolve
File: tfamily merger_trees.evolve.F90
Default value: false
Description: Specifies whether merger tree structure should be dumped to a DOT file.

Name: tfamily mergerTreesWrite
Attached to: tfamily module:Merger_Trees_Write
File: tfamily merger_trees.write.F90
Default value: false
Description: Specifies whether or not trees should be written to file.

Name: tfamily mergingTimescaleMultiplier
Attached to: tfamily module:Dynamical_Friction_Timescale_Utilities
File: tfamily satellites.merging.timescale.utilities.F90
Default value: 0.75
Description: A multiplier for the merging timescale in dynamical friction timescale calculations.

Name: tfamily metaCollectTimingData
Attached to: tfamily module:Galacticus_Meta_Tree_Timing
File: tfamily galacticus.meta.tree_timing.F90
Default value: false
Description: Specifies whether or not collect and output data on the time spent processing trees.

Name: tfamily metaProfileTimeStepMaximum
Attached to: tfamily module:Galacticus_Meta_Evolver_Profiler
File: tfamily galacticus.meta.evolver_profiler.F90
Default value: 10 Gyr
Description: The largest timestep to use in profiling ODE solver steps.
Name: tfamily metaProfileTimeStepMinimum
Attached to: tfamily module:Galacticus_Meta_Evolver_Profiler
File: tfamily galacticus.meta.evolver_profiler.F90
Default value: $10^{-6}$ Gyr
Description: The smallest timestep to use in profiling ODE solver steps.

Name: tfamily metaProfileTimeStepPointsPerDecade
Attached to: tfamily module:Galacticus_Meta_Evolver_Profiler
File: tfamily galacticus.meta.evolver_profiler.F90
Default value: 3
Description: The number of bins per decade of timestep to use when profiling ODE solver steps.

Name: tfamily minorMergerGasMovesTo
Attached to: tfamily module:Satellite_Merging_Mass_Movements_Simple
File: tfamily satellites.merging.mass_movements.simple.F90
Default value: spheroid
Description: The component to which satellite galaxy gas moves to as a result of a minor merger.

Name: tfamily modifiedPressSchechterFirstOrderAccuracy
Attached to: tfamily module:Modified_Press_Schechter_Branching
File: tfamily merger_trees.branching_probability.modified_Press_Schechter.F90
Default value: 0.1
Description: Limits the step in $\delta_{\text{crit}}$ when constructing merger trees using the Parkinson et al. [2008] algorithm, so that it never exceeds $\text{modifiedPressSchechterFirstOrderAccuracy} \sqrt{2[\sigma^2(M_2/2) - \sigma^2(M_2)]}$.

Name: tfamily modifiedPressSchechterG0
Attached to: tfamily module:Merger_Tree_Branching_Modifiers_Parkinson
File: tfamily merger_trees.branching_probability.modifier.Parkinson.F90
Default value: 0.57
Description: The parameter $G_0$ appearing in the modified merger rate expression of Parkinson et al. [2008].

Name: tfamily modifiedPressSchechterGamma1
Attached to: tfamily module:Merger_Tree_Branching_Modifiers_Parkinson
File: tfamily merger_trees.branching_probability.modifier.Parkinson.F90
Default value: 0.38
Description: The parameter $\gamma_1$ appearing in the modified merger rate expression of Parkinson et al. [2008].

Name: tfamily modifiedPressSchechterGamma2
Attached to: tfamily module:Merger_Tree_Branching_Modifiers_Parkinson
File: tfamily merger_trees.branching_probability.modifier.Parkinson.F90
Default value: -0.01
Description: The parameter $\gamma_2$ appearing in the modified merger rate expression of Parkinson et al. [2008].

Name: tfamily molecularComplexClumpingFactorKMT09
Attached to: tfamily module:Star_Formation_Rate_Surface_Density_Disks_KMT09
File: tfamily star_formation.rate_surface_density.disks.KMT09.F90
4. Input Parameters

**Default value:** 5  
**Description:** The density enhancement (relative to mean disk density) for molecular complexes in the “Krumholz-McKee-Tumlinson” star formation timescale calculation.

**Name:** tfamily molecularFractionFastKMT09  
**Attached to:** tfamily module:Star_Formation_Rate_Surface_Density_Disks_KMT09  
**File:** tfamily star_formation.rate.surface_density.disks.KMT09.F90  
**Default value:** true  
**Description:** Selects whether the fast (but less accurate) fitting formula for molecular hydrogen should be used in the “Krumholz-McKee-Tumlinson” star formation timescale calculation.

**Name:** tfamily nfw1996ConcentrationC  
**Attached to:** tfamily module:Dark_Matter_Profiles_Concentration  
**File:** tfamily dark_matter_profiles.structure.concentration.F90  
**Default value:** 2000 Navarro et al. [1996]  
**Description:** The parameter $f$ appearing in the halo concentration algorithm of Navarro et al. [1996].

**Name:** tfamily nfw1996ConcentrationF  
**Attached to:** tfamily module:Dark_Matter_Profiles_Concentration  
**File:** tfamily dark_matter_profiles.structure.concentration.F90  
**Default value:** 0.01 Navarro et al. [1996]  
**Description:** The parameter $C$ appearing in the halo concentration algorithm of Navarro et al. [1996].

**Name:** tfamily nodeFormationMassFraction  
**Attached to:** tfamily module:Node_Component_Merging_Statistics_Standard  
**File:** tfamily objects.nodes.components.merging_statistics.standard.F90  
**Default value:** 0.5  
**Description:** The mass fraction in the main branch progenitor used to define the formation time of each halo.

**Name:** tfamily nodeMajorMergerFraction  
**Attached to:** tfamily module:Node_Component_Merging_Statistics_Recent  
**File:** tfamily objects.nodes.components.merging_statistics.recent.F90  
**Default value:** 0.25  
**Description:** The mass ratio ($M_2/M_1$ where $M_2 < M_1$) of merging halos above which the merger should be considered to be “major”.

**Name:** tfamily nodeMergersMethod  
**Attached to:** tfamily module:Merger_Trees_Evolve_Node  
**File:** tfamily merger_trees.evolve.node.F90  
**Default value:** singleLevelHierarchy  
**Description:** Selects the method to be used for handling node merger events.

**Name:** tfamily nodePromotionIndexShift  
**Attached to:** tfamily module:Node_Promotion_Index_Shifts  
**File:** tfamily events.node_promotion.index.shift.F90  
**Default value:** false  
**Description:** Specifies whether or not the index of a node should be shifted to its parent node prior to promotion.
Name: tfamily nodeRecentMajorMergerFromInfall  
Attached to: tfamily module:Node_Component_Merging_Statistics_Recent  
File: tfamily objects.nodes.components.merging_statistics.recent.F90  
Default value: false  
Description: Specifies whether “recent” for satellite galaxies is measured from the current time, or from the time at which they were last isolated.

Name: tfamily nodeRecentMajorMergerInterval  
Attached to: tfamily module:Node_Component_Merging_Statistics_Recent  
File: tfamily objects.nodes.components.merging_statistics.recent.F90  
Default value: 2.0  
Description: The time interval used to define “recent” mergers in the recent merging statistics component. This parameter is in units of Gyr if [nodeRecentMajorMergerIntervalType]=absolute, or in units of the halo dynamical time if [nodeRecentMajorMergerIntervalType]=dynamical.

Name: tfamily nodeRecentMajorMergerIntervalType  
Attached to: tfamily module:Node_Component_Merging_Statistics_Recent  
File: tfamily objects.nodes.components.merging_statistics.recent.F90  
Default value: dynamical  
Description: Specifies the units for the [nodeRecentMajorMergerInterval] parameter. If set to absolute then [nodeRecentMajorMergerInterval] is given in Gyr, while if set to dynamical [nodeRecentMajorMergerInterval] is given in units of the halo dynamical time.

Name: tfamily noninstantHistoryTimesCount  
Attached to: tfamily module:Stellar_Population_Properties_Noninstantaneous  
File: tfamily stellar_populations.properties.noninstantaneous.F90  
Default value: 10  
Description: The number of times at which a galaxy’s stellar properties history is stored.

Name: tfamily odeAlgorithm  
Attached to: tfamily module:Merger_Trees_Evolve_Node  
File: tfamily merger_trees.evolve.node.F90  
Default value: Runge-Kutta-Cash-Karp  
Description: The algorithm to use in the ODE solver.

Name: tfamily odeToleranceAbsolute  
Attached to: tfamily module:Merger_Trees_Evolve_Node  
File: tfamily merger_trees.evolve.node.F90  
Default value: 0.01  
Description: The absolute tolerance used in solving differential equations for node evolution.

Name: tfamily odeToleranceRelative  
Attached to: tfamily module:Merger_Trees_Evolve_Node  
File: tfamily merger_trees.evolve.node.F90  
Default value: 0.01  
Description: The relative tolerance used in solving differential equations for node evolution.

Name: tfamily optimalSamplingDensityOutputFileName
4. Input Parameters

**Attached to:** tfamily program:Optimal_Sampling_SMF  
**File:** tfamily optimal_sampling.stellar_mass_function.F90  
**Default value:** none  
**Description:** The name of a file to which the optimal tree mass sampling density function will be written.

**Name:** tfamily optimalSamplingLogarithmicBinWidth  
**Attached to:** tfamily program:Optimal_Sampling_SMF  
**File:** tfamily optimal_sampling.stellar_mass_function.F90  
**Default value:** none  
**Description:** The logarithmic width of bins in the stellar mass function to use when computing the optimal tree mass sampling density function.

**Name:** tfamily outputColdModeInfallRate  
**Attached to:** tfamily module:Cooling_Cold_Mode_Infall_Rates  
**File:** tfamily cooling.cold_mode.infall_rate.F90  
**Default value:** false  
**Description:** Determines whether or not cold mode infall rates are output.

**Name:** tfamily outputDensityContrastData  
**Attached to:** tfamily module:Galacticus_Output_Trees_Density_Contrasts  
**File:** tfamily galacticus.output.merger_tree.density_contrasts.F90  
**Default value:** false  
**Description:** Specifies whether or not density contrast data (i.e. radius and mass at a given density contrast) should be included in the output.

**Name:** tfamily outputDensityContrastDataDarkOnly  
**Attached to:** tfamily module:Galacticus_Output_Trees_Density_Contrasts  
**File:** tfamily galacticus.output.merger_tree.density_contrasts.F90  
**Default value:** false  
**Description:** Specifies whether or not density contrast data should be computed using the dark matter component alone.

**Name:** tfamily outputDensityContrastHaloLoaded  
**Attached to:** tfamily module:Galacticus_Output_Trees_Density_Contrasts  
**File:** tfamily galacticus.output.merger_tree.density_contrasts.F90  
**Default value:** true  
**Description:** Specifies whether baryonic loading of the halo should be accounted for when outputting density contrast data.

**Name:** tfamily outputDensityContrastValues  
**Attached to:** tfamily module:Galacticus_Output_Trees_Density_Contrasts  
**File:** tfamily galacticus.output.merger_tree.density_contrasts.F90  
**Default value:** none  
**Description:** A list of density contrasts at which to output data.

**Name:** tfamily outputDescendentIndices  
**Attached to:** tfamily module:Galacticus_Output_Trees_Descendents  
**File:** tfamily galacticus.output.merger_tree.descendents.F90  
**Default value:** false
**Description:** Specifies whether or not descendent indices (i.e. index of the node at the next output) should be included in the output.

**Name:** tfamily outputFinalDescendentIndices  
**Attached to:** tfamily module:Galacticus_Output_Trees_Final_Descendents  
**File:** tfamily galacticus.output.merger_tree.final_descendent.F90  
**Default value:** false  
**Description:** Specifies whether or not final descendent indices (i.e. index of the node at the base of the tree) should be included in the output.

**Name:** tfamily outputHalfLightData  
**Attached to:** tfamily module:Galacticus_Output_Tree_Half_Light_Properties  
**File:** tfamily galacticus.output.merger_tree.half_light_properties.F90  
**Default value:** false  
**Description:** Specifies whether or not half-light radius data (i.e. radius and mass) should be included in the output.

**Name:** tfamily outputHalfMassData  
**Attached to:** tfamily module:Galacticus_Output_Tree_Half_Mass_Radii  
**File:** tfamily galacticus.output.merger_tree.half_mass_radius.F90  
**Default value:** false  
**Description:** Specifies whether or not half-mass radii should be included in the output.

**Name:** tfamily outputHaloAccretionMode  
**Attached to:** tfamily module:Accretion_Halos  
**File:** tfamily accretion.halo.F90  
**Default value:** false  
**Description:** Determines whether or not halo accretion rates are output.

**Name:** tfamily outputHaloModelData  
**Attached to:** tfamily module:Galacticus_Output_Halo_Models  
**File:** tfamily galacticus.output.merger_tree.halo_model.F90  
**Default value:** false  
**Description:** Specifies whether or not halo model data (bias, power spectra, etc.) should be included in the output.

**Name:** tfamily outputHotHaloCoolingRadii  
**Attached to:** tfamily module:Cooling_Radii  
**File:** tfamily cooling.cooling_radius.F90  
**Default value:** false  
**Description:** Determines whether or not cooling radii are output.

**Name:** tfamily outputHotHaloCoolingRates  
**Attached to:** tfamily module:Cooling_Rates  
**File:** tfamily cooling.cooling_rate.F90  
**Default value:** false  
**Description:** Determines whether or not cooling rates and radii are output.

**Name:** tfamily outputMainBranchStatus
4. Input Parameters

**Attached to:** tfamily module:Galacticus_Output_Trees_Main_Branch
**File:** tfamily galacticus.output.merger_tree.main_branch.F90
**Default value:** false
**Description:** Controls whether or not the main branch status of each node will be output.

**Name:** tfamily outputMassProfileData
**Attached to:** tfamily module:Galacticus_Output_Tree_Mass_Profiles
**File:** tfamily galacticus.output.merger_tree.mass_profile.F90
**Default value:** false
**Description:** Specifies whether or not half-light radius data (i.e. radius and mass) should be included in the output.

**Name:** tfamily outputMassProfileRadii
**Attached to:** tfamily module:Galacticus_Output_Tree_Mass_Profiles
**File:** tfamily galacticus.output.merger_tree.mass_profile.F90
**Default value:** none
**Description:** A list of radii at which to output the mass profile.

**Name:** tfamily outputMostMassiveProgenitor
**Attached to:** tfamily module:Galacticus_Output_Most_Massive_Progenitors
**File:** tfamily galacticus.output.merger_tree.most_massive_progenitor.F90
**Default value:** false
**Description:** Specifies whether or not most massive progenitor status should be output.

**Name:** tfamily outputNodeRedshifts
**Attached to:** tfamily module:Galacticus_Output_Trees_Redshifts
**File:** tfamily galacticus.output.merger_tree.redshifts.F90
**Default value:** false
**Description:** Controls whether or not the redshifts corresponding to node times should be output.

**Name:** tfamily outputRedshifts
**Attached to:** tfamily module:Halo_Mass_Function_Tasks
**File:** tfamily halo_mass_functions.tasks.F90
**Default value:** 0
**Description:** A list of redshifts at which halo mass functions should be computed.

**Name:** tfamily outputRotationCurveData
**Attached to:** tfamily module:Galacticus_Output_Trees_Rotation_Curve
**File:** tfamily galacticus.output.merger_tree.rotation_curve.F90
**Default value:** false
**Description:** Specifies whether or not rotation curve data should be included in the output file.

**Name:** tfamily outputRotationCurveRadii
**Attached to:** tfamily module:Galacticus_Output_Trees_Rotation_Curve
**File:** tfamily galacticus.output.merger_tree.rotation_curve.F90
**Default value:** none
**Description:** Specifies the radii at which the rotation curve will be output.

**Name:** tfamily outputSatelliteApocenterData
Attached to: tfamily module:Galacticus_Output_Trees_Satellite_Extremum
File: tfamily galacticus.output.merger_tree.satellite_extrema.F90
Default value: false
Description: Specifies whether or not satellite orbital apocenter data (radius, velocity) should be included in the output.

Name: tfamily outputSatelliteHostData
Attached to: tfamily module:Galacticus_Output_Trees_Satellite_Host
File: tfamily galacticus.output.merger_tree.satellite_host.F90
Default value: false
Description: Specifies whether or not satellite host data (node mass) should be included in the output.

Name: tfamily outputSatellitePericenterData
Attached to: tfamily module:Galacticus_Output_Trees_Satellite_Extremum
File: tfamily galacticus.output.merger_tree.satellite_extrema.F90
Default value: false
Description: Specifies whether or not satellite orbital pericenter data (radius, velocity) should be included in the output.

Name: tfamily outputSatelliteStatus
Attached to: tfamily module:Galacticus_Output_Trees_Satellite_Extremum
File: tfamily galacticus.output.merger_tree.satellite_status.F90
Default value: false
Description: Specifies whether or not satellite status (i.e., whether the satellite is orphaned or not) should be included in the output.

Name: tfamily outputTreeIndices
Attached to: tfamily module:Galacticus_Output_Trees_Tree_Indices
File: tfamily galacticus.output.merger_tree.tree_indices.F90
Default value: false
Description: Specifies whether or not descendent indices (i.e., index of the node at the next output) should be included in the output.

Name: tfamily outputVelocityDispersionData
Attached to: tfamily module:Galacticus_Output_Trees_Velocity_Dispersion
File: tfamily galacticus.output.merger_tree.velocity_dispersion.F90
Default value: false
Description: Specifies whether or not velocity dispersion data should be included in the output file.

Name: tfamily outputVelocityDispersionRadii
Attached to: tfamily module:Galacticus_Output_Trees_Velocity_Dispersion
File: tfamily galacticus.output.merger_tree.velocity_dispersion.F90
Default value: none
Description: Specifies the radii at which the velocity dispersion will be output.

Name: tfamily outputVirialData
Attached to: tfamily module:Galacticus_Output_Trees_Virial
File: tfamily galacticus.output.merger_tree.virial.F90
Default value: false
4. Input Parameters

**Description:** Specifies whether or not virial data (radius, velocity) should be included in the output.

**Name:** tfamily powerSpectraPointsPerDecade  
**Attached to:** tfamily module:Power_Spectrum_Tasks  
**File:** tfamily power_spectra.tasks.F90  
**Default value:** 10  
**Description:** The number of points per decade of wavenumber at which to tabulate power spectra.

**Name:** tfamily powerSpectraWavenumberMaximum  
**Attached to:** tfamily module:Power_Spectrum_Tasks  
**File:** tfamily power_spectra.tasks.F90  
**Default value:** $10^3$ Mpc$^{-1}$  
**Description:** The maximum wavenumber at which to tabulate power spectra.

**Name:** tfamily powerSpectraWavenumberMinimum  
**Attached to:** tfamily module:Power_Spectrum_Tasks  
**File:** tfamily power_spectra.tasks.F90  
**Default value:** $10^{-3}$ Mpc$^{-1}$  
**Description:** The minimum wavenumber at which to tabulate power spectra.

**Name:** tfamily powerSpectrumIndex  
**Attached to:** tfamily module:Merger_Trees_Simple  
**File:** tfamily merger_trees.file_maker.simple.F90  
**Default value:** none  
**Description:** The index of the power-law primordial power spectrum.

**Name:** tfamily powerSpectrumMethod  
**Attached to:** tfamily module:Primordial_Power_Spectra  
**File:** tfamily structure_formation.power_spectrum.primordial.F90  
**Default value:** powerLaw  
**Description:** The name of the method to be used for computing the primordial power spectrum.

**Name:** tfamily powerSpectrumNonlinearMethod  
**Attached to:** tfamily module:Power_Spectra_Nonlinear  
**File:** tfamily structure_formation.power_spectrum.nonlinear.F90  
**Default value:** CosmicEmu  
**Description:** The name of the method to be used for computing the nonlinear power spectrum.

**Name:** tfamily powerSpectrumReferenceWavenumber  
**Attached to:** tfamily module:Primordial_Power_Spectrum_Power_Law  
**File:** tfamily structure_formation.power_spectrum.primordial.power_law.F90  
**Default value:** 1  
**Description:** When a running power spectrum index is used, this is the wavenumber at which the index is equal to [powerSpectrumIndex].

**Name:** tfamily powerSpectrumRunning  
**Attached to:** tfamily module:Primordial_Power_Spectrum_Power_Law  
**File:** tfamily structure_formation.power_spectrum.primordial.power_law.F90  
**Default value:** 0
**Description:** The running, $dn_s/d\ln k$, of the power spectrum index.

**Name:** tfamily powerSpectrumWindowFunctionMethod  
**Attached to:** tfamily module:Power_Spectrum_Window_Functions  
**File:** tfamily structure_formation.power_spectrum.variance.window_function.F90  
**Default value:** topHat  
**Description:** The name of the method to be used for computing window functions to estimate variance from the power spectrum.

**Name:** tfamily powerSpectrumWindowFunctionSharpKSpaceNormalization  
**Attached to:** tfamily module:Power_Spectrum_Window_Functions_TH_KSS_Hybrid  
**File:** tfamily structure_formation.power_spectrum.variance.window_function.top_hat_kspace_sharp_.hybrid.F90  
**Default value:** natural  
**Description:** The parameter $a$ in the relation $k_s = a/r_s$, where $k_s$ is the cut-off wavenumber for the sharp $k$-space window function and $r_s$ is the radius of a sphere (in real-space) enclosing the requested smoothing mass. Alternatively, a value of `natural` will be supplied in which case the normalization is chosen such that, in real-space, $W(r = 0) = 1$. This results in a contained mass of $M = 6\pi^2\bar{\rho}k_s^{-3}$.

**Name:** tfamily powerSpectrumWindowFunctionSharpKSpaceTopHatRadiusRatio  
**Attached to:** tfamily module:Power_Spectrum_Window_Functions_TH_KSS_Hybrid  
**File:** tfamily structure_formation.power_spectrum.variance.window_function.top_hat_kspace_sharp_.hybrid.F90  
**Default value:** 1  
**Description:** The parameter $\beta$ in the relation $r_s = \beta r_{th}$ between $k$-space sharp and top-hat window function radii in the hybrid window function used for computing the variance in the power spectrum.

**Name:** tfamily prada2011ConcentrationA  
**Attached to:** tfamily module:Dark_Matter_Profiles_Concentration  
**File:** tfamily dark_matter_profiles.structure.concentration.F90  
**Default value:** 2.881 Prada et al. [2011]  
**Description:** The parameter $A$ appearing in the halo concentration algorithm of Prada et al. [2011].

**Name:** tfamily prada2011ConcentrationAlpha  
**Attached to:** tfamily module:Dark_Matter_Profiles_Concentration  
**File:** tfamily dark_matter_profiles.structure.concentration.F90  
**Default value:** 6.948 Prada et al. [2011]  
**Description:** The parameter $\alpha$ appearing in the halo concentration algorithm of Prada et al. [2011].

**Name:** tfamily prada2011ConcentrationBeta  
**Attached to:** tfamily module:Dark_Matter_Profiles_Concentration  
**File:** tfamily dark_matter_profiles.structure.concentration.F90  
**Default value:** 7.386 Prada et al. [2011]  
**Description:** The parameter $b$ appearing in the halo concentration algorithm of Prada et al. [2011].
4. Input Parameters

**Description:** The parameter $\beta$ appearing in the halo concentration algorithm of Prada et al. [2011].

**Name:** tfamily prada2011ConcentrationC
**Attached to:** tfamily module:Dark_Matter_Profiles_Concentration
**File:** tfamily dark_matter_profiles.structure.concentration.F90
**Default value:** 1.022 Prada et al. [2011]
**Description:** The parameter $c$ appearing in the halo concentration algorithm of Prada et al. [2011].

**Name:** tfamily prada2011ConcentrationC0
**Attached to:** tfamily module:Dark_Matter_Profiles_Concentration
**File:** tfamily dark_matter_profiles.structure.concentration.F90
**Default value:** 3.681 Prada et al. [2011]
**Description:** The parameter $c_0$ appearing in the halo concentration algorithm of Prada et al. [2011].

**Name:** tfamily prada2011ConcentrationC1
**Attached to:** tfamily module:Dark_Matter_Profiles_Concentration
**File:** tfamily dark_matter_profiles.structure.concentration.F90
**Default value:** 5.033 Prada et al. [2011]
**Description:** The parameter $c_1$ appearing in the halo concentration algorithm of Prada et al. [2011].

**Name:** tfamily prada2011ConcentrationD
**Attached to:** tfamily module:Dark_Matter_Profiles_Concentration
**File:** tfamily dark_matter_profiles.structure.concentration.F90
**Default value:** 0.060 Prada et al. [2011]
**Description:** The parameter $d$ appearing in the halo concentration algorithm of Prada et al. [2011].

**Name:** tfamily prada2011ConcentrationInverseSigma0
**Attached to:** tfamily module:Dark_Matter_Profiles_Concentration
**File:** tfamily dark_matter_profiles.structure.concentration.F90
**Default value:** 1.047 Prada et al. [2011]
**Description:** The parameter $\sigma^{-1}_0$ appearing in the halo concentration algorithm of Prada et al. [2011].

**Name:** tfamily prada2011ConcentrationInverseSigma1
**Attached to:** tfamily module:Dark_Matter_Profiles_Concentration
**File:** tfamily dark_matter_profiles.structure.concentration.F90
**Default value:** 1.646 Prada et al. [2011]
**Description:** The parameter $\sigma^{-1}_1$ appearing in the halo concentration algorithm of Prada et al. [2011].

**Name:** tfamily prada2011ConcentrationX0
**Attached to:** tfamily module:Dark_Matter_Profiles_Concentration
**File:** tfamily dark_matter_profiles.structure.concentration.F90
**Default value:** 0.424 Prada et al. [2011]
**Description:** The parameter $x_0$ appearing in the halo concentration algorithm of Prada et al. [2011].

**Name:** tfamily prada2011ConcentrationX1
**Attached to:** tfamily module:Dark_Matter_Profiles_Concentration
**File:** tfamily dark_matter_profiles.structure.concentration.F90
**Default value:** 0.526 Prada et al. [2011]
**Description:** The parameter $x_1$ appearing in the halo concentration algorithm of Prada et al. [2011].
Name: tfamily pressureCharacteristicBlitzRosolowsky
Attached to: tfamily module:Star_Formation_Rate_Surface_Density_Disks_BR
File: tfamily star_formation.rate_surface_density.disks.Blitz-Rosolowsky.F90
Default value: 4.54 [Blitz and Rosolowsky, 2006]
Description: The characteristic pressure (given as \( P_0/k_B \) in units of K cm\(^{-3}\)) in the scaling relation of molecular hydrogen fraction with disk pressure in the “Blitz-Rosolowsky” star formation timescale calculation.

Name: tfamily pressureExponentBlitzRosolowsky
Attached to: tfamily module:Star_Formation_Rate_Surface_Density_Disks_BR
File: tfamily star_formation.rate_surface_density.disks.Blitz-Rosolowsky.F90
Default value: 0.92 [Blitz and Rosolowsky, 2006]
Description: The exponent in the scaling relation of molecular hydrogen fraction with disk pressure in the “Blitz-Rosolowsky” star formation timescale calculation.

Name: tfamily profileOdeEvolver
Attached to: tfamily module:Merger_Trees_Evolve_Node
File: tfamily merger_trees.evolve.node.F90
Default value: false
Description: Specifies whether or not to profile the ODE evolver.

Name: tfamily radiationIGBFileName
Attached to: tfamily module:Radiation_IGB_File
File: tfamily radiation.intergalactic_background.file.F90
Default value: none
Description: The name of the file containing a tabulation of the radiation field.

Name: tfamily radiationIntergalacticBackgroundMethod
Attached to: tfamily module:Radiation_Intergalactic_Background
File: tfamily radiation.intergalactic_background.F90
Default value: file
Description: The name of the method to be used for calculations of the intergalactic background radiation field.

Name: tfamily ramPressureStrippingFormFactor
Attached to: tfamily module:Hot_Halo_Ram_Pressure_Stripping_Font2008
Default value: 2
Description: The form factor appearing in the gravitational binding force (per unit area) in the ram pressure stripping model of Font et al. (2008; their eqn. 4).

Name: tfamily ramPressureStrippingMassLossRateDiskSimpleFractionalRateMaximum
Attached to: tfamily module:Ram_Pressure_Stripping_Mass_Loss_Rate_Disks_Simple
File: tfamily ram_pressure_stripping.mass_loss_rate.disks.simple.F90
Default value: 10
Description: The maximum fractional mass loss rate per dynamical time in the simple model of mass loss from disks due to ram pressure stripping.
4. Input Parameters

Name: tfamily ramPressureStrippingMassLossRateDisksMethod
Attached to: tfamily module:Ram_Pressure_Stripping_Mass_Loss_Rate_Disks
File: tfamily ram_pressure_stripping.mass_loss_rate.disks.F90
Default value: null
Description: The name of the method to be used when computing mass loss rates from disks due to ram pressure stripping.

Name: tfamily ramPressureStrippingMassLossRateSpheroidSimpleFractionalRateMaximum
Attached to: tfamily module:Ram_Pressure_Stripping_Mass_Loss_Rate_Spheroids_Simple
File: tfamily ram_pressure_stripping.mass_loss_rate.spheroids.simple.F90
Default value: 10
Description: The maximum fractional mass loss rate per dynamical time in the simple model of mass loss from spheroids due to ram pressure stripping.

Name: tfamily ramPressureStrippingMassLossRateSpheroidsMethod
Attached to: tfamily module:Ram_Pressure_Stripping_Mass_Loss_Rate_Spheroids
File: tfamily ram_pressure_stripping.mass_loss_rate.spheroids.F90
Default value: null
Description: The name of the method to be used when computing mass loss rates from spheroids due to ram pressure stripping.

Name: tfamily randomSeed
Attached to: tfamily module:Pseudo_Random
File: tfamily numerical.random.F90
Default value: 219
Description: A seed value for the random number generator.

Name: tfamily randomSpinResetMassFactor
Attached to: tfamily module:Node_Component_Spin_Random
File: tfamily objects.nodes.components.spin.random.F90
Default value: 2.0
Description: The factor by which a node must increase in mass before its spin parameter is reset.

Name: tfamily reionizationSuppressionOpticalDepth
Attached to: tfamily module:Accretion_Halos
File: tfamily accretion.halo.F90
Default value: none
Description: The optical depth to electron scattering below which baryonic accretion is suppressed.

Name: tfamily reionizationSuppressionRedshift
Attached to: tfamily module:Accretion_Halos
File: tfamily accretion.halo.F90
Default value: 9.97 (Hinshaw et al. 2012; CMB+$H_0$+BAO)
Description: The redshift below which baryonic accretion is suppressed.

Name: tfamily reionizationSuppressionVelocity
Attached to: tfamily module:Accretion_Halos
File: tfamily accretion.halo.F90
Default value: 35.0
**Description:** The velocity scale below which baryonic accretion is suppressed.

**Name:** tfamily satelliteDynamicalFrictionChandrasekharCoulombLogarithm  
**Attached to:** tfamily module:Dynamical_Friction_Acceleration_Chandrasekhar  
**File:** tfamily satellites.dynamical_friction.acceleration.Chandrasekhar1943.F90  
**Default value:** 2  
**Description:** The Coulomb logarithm, ln Λ, appearing in the Chandrasekhar [1943] formulation of the acceleration due to dynamical friction.

**Name:** tfamily satelliteDynamicalFrictionMethod  
**Attached to:** tfamily module:Satellite_Dynamical_Friction  
**File:** tfamily satellites.dynamical_friction.acceleration.F90  
**Default value:** Chandrasekhar1943  
**Description:** The name of the method to be used to compute satellite dynamical friction acceleration.

**Name:** tfamily satelliteMergingJiang2008Scatter  
**Attached to:** tfamily module:Satellite_Merging_Timescales  
**File:** tfamily satellites.merging.timescale.F90  
**Default value:** 0  
**Description:** Specifies whether or not to add random scatter to the dynamical friction timescales in the Jiang2008 satellite merging time implementation.

**Name:** tfamily satelliteMergingMassMovementsMethod  
**Attached to:** tfamily module:Satellite_Merging_Mass_Movements  
**File:** tfamily satellites.merging.mass_movements.F90  
**Default value:** simple  
**Description:** Selects the method to be used for deciding mass movements during satellite mergers.

**Name:** tfamily satelliteMergingRemnantProgenitorPropertiesMethod  
**Attached to:** tfamily module:Satellite_Merging_Remnant_Sizes_Progenitors  
**File:** tfamily satellites.merging.remnant_sizes.progenitor_properties.F90  
**Default value:** standard  
**Description:** The name of the method to be used for computing progenitor properties in merger remnant calculations.

**Name:** tfamily satelliteMergingRemnantSizeMethod  
**Attached to:** tfamily module:Satellite_Merging_Remnant_Sizes  
**File:** tfamily satellites.merging.remnant_sizes.F90  
**Default value:** Covington2008  
**Description:** The name of the method to be used for computing merger remnant sizes.

**Name:** tfamily satelliteMergingTimescaleVillalobos2013BaseMethod  
**Attached to:** tfamily module:Satellite_Merging_Timescales  
**File:** tfamily satellites.merging.timescale.F90  
**Default value:** boylanKolchin2008  
**Description:** The base satelliteMergingTimescales method to which the Villalobos et al. [2013] modifier for satellite merging timescales should be applied.
4. Input Parameters

**Attached to:** tfamily module: Satellite_Merging_Timescales  
**File:** tfamily satellites.merging.timescale.F90  
**Default value:** 0.44 [Villalobos et al., 2013]  
**Description:** The exponent of $1 + z$ appearing in the Villalobos et al. [2013] modifier for satellite merging timescales.

**Name:** tfamily satelliteMergingTimescalesMethod  
**Attached to:** tfamily module: Satellite_Merging_Timescales  
**File:** tfamily satellites.merging.timescale.F90  
**Default value:** jiang2008  
**Description:** The method to be used for satelliteMergingTimescales.

**Name:** tfamily satelliteOrbitResetOnHaloFormation  
**Attached to:** tfamily module: Node_Component_Satellite_Very_Simple  
**File:** tfamily objects.nodes.components.satellite.very_simple.F90  
**Default value:** false  
**Description:** Specifies whether satellite virial orbital parameters should be reset on halo formation events.

**Name:** tfamily satelliteOrbitStoreOrbitalParameters  
**Attached to:** tfamily module: Node_Component_Satellite_Standard  
**File:** tfamily objects.nodes.components.satellite.standard.F90  
**Default value:** true  
**Description:** Specifies whether satellite virial orbital parameters should be stored (otherwise they are computed again—possibly at random—each time they are requested).

**Name:** tfamily satelliteOrbitingDestructionMassFraction  
**Attached to:** tfamily module: Node_Component_Satellite_Orbiting  
**File:** tfamily objects.nodes.components.satellite.orbiting.F90  
**Default value:** 0.01  
**Description:** The fraction of the satellite’s initial mass below which the satellite is considered to be tidally destroyed and merged with the central halo.

**Name:** tfamily satelliteOutputVirialOrbit  
**Attached to:** tfamily module: Galacticus_Nodes  
**File:** tfamily objects.nodes.F90  
**Default value:** false  
**Description:** Specifies whether the virialOrbit method of the standard implementation of the satellite component class should be output.

**Name:** tfamily satelliteTidalFieldBoostFactor  
**Attached to:** tfamily module: Satellites_Tidal_Fields_Spherical_Symmetry  
**File:** tfamily satellites.tidal_fields.spherical_symmetry.F90  
**Default value:** 1.0  
**Description:** The factor by which to boost satellite tidal fields in the sphericalSymmetry tidal field method.

**Name:** tfamily satelliteTidalHeatingGnedinEpsilon  
**Attached to:** tfamily module: Tidal_Heating_Rate_Gnedin
Default value: 3
Description: Parameter, $\epsilon$, controlling the tidal heating rate of satellites in the Gnedin1999 method.

Name: tfamily satelliteTidalHeatingGnedinGamma
Attached to: tfamily module:Tidal_Heating_Rate_Gnedin
Default value: 2.5
Description: Parameter, $\gamma$, controlling the tidal heating rate of satellites in the Gnedin1999 method.

Name: tfamily satelliteTidalHeatingMethod
Attached to: tfamily module:Satellite_Tidal_Heating
File: tfamily satellites.tidal_heating.rate.F90
Default value: Gnedin1999
Description: The name of the method to be used to compute satellite tidal heating rate.

Name: tfamily satelliteTidalStrippingMethod
Attached to: tfamily module:Satellite_Tidal_Stripping
File: tfamily satellites.tidal_stripping.rate.F90
Default value: Zentner2005
Description: The name of the method to be used to compute satellite tidal stripping rate.

Name: tfamily satelliteTidalStrippingZentner2005Rate
Attached to: tfamily module:Tidal_Stripping_Rate_Zentner2005
Default value: 2.0
Description: The dimensionless rate coefficient appearing in the Zentner et al. [2005] expression for the tidal mass loss rate from subhalos.

Name: tfamily satellitesTidalFieldMethod
Attached to: tfamily module:Satellites_Tidal_Fields
File: tfamily satellites.tidal_fields.F90
Default value: null
Description: The name of the method to be used when computing the tidal field acting on a satellite.

Name: tfamily sigma_8
Attached to: tfamily module:Power_Spectra
File: tfamily structure_formation.power_spectrum.F90
Default value: 0.817 (Hinshaw et al. 2012; CMB+$H_0$+BAO)
Description: The fractional mass fluctuation in the linear density field at the present day in spheres of radius 8 Mpc/h.

Name: tfamily simpleRadiusSolverUseFormationHalo
Attached to: tfamily module:Galactic_Structure_Radii_Simple
File: tfamily galactic_structure.radius_solver.simple.F90
Default value: false
Description: Specifies whether or not the “formation halo” should be used when solving for the radii of galaxies.
4. Input Parameters

Name: tfamily source
Attached to: tfamily module:Merger_Trees_Simple
File: tfamily merger_trees.file_maker.simple.F90
Default value: none
Description: The source of the merger trees.

Name: tfamily spectraPostprocessorMethod
Attached to: tfamily module:Stellar_Population_Spectra_Postprocess
File: tfamily stellar_populations.spectra.postprocess.F90
Default value: null
Description: The method to be used for spectraPostprocessor.

Name: tfamily spheroidAngularMomentumAtScaleRadius
Attached to: tfamily module:Node_Component_Spheroid_Standard
File: tfamily objects.nodes.components.spheroid.standard.F90
Default value: $I_2/I_3$ where $I_n = \int_0^\infty \rho(r)r^n dr$, where $\rho(r)$ is the spheroid density profile, unless either $I_2$ or $I_3$ is infinite, in which case a default of $1/2$ is used instead
Description: The assumed ratio of the specific angular momentum at the scale radius to the mean specific angular momentum of the standard spheroid component.

Name: tfamily spheroidEnergeticOutflowMassRate
Attached to: tfamily module:Node_Component_Spheroid_Standard
File: tfamily objects.nodes.components.spheroid.standard.F90
Default value: 0.01
Description: The proportionality factor relating mass outflow rate from the spheroid to the energy input rate divided by $V_{spheroid}^2$.

Name: tfamily spheroidMassDistribution
Attached to: tfamily module:Node_Component_Spheroid_Standard
File: tfamily objects.nodes.components.spheroid.standard.F90
Default value: hernquist
Description: The type of mass distribution to use for the standard spheroid component.

Name: tfamily spheroidMassToleranceAbsolute
Attached to: tfamily module:Node_Component_Spheroid_Standard
File: tfamily objects.nodes.components.spheroid.standard.F90
Default value: $10^{-6} M_\odot$
Description: The mass tolerance used to judge whether the spheroid is physically plausible.

Name: tfamily spheroidOutflowExponent
Attached to: tfamily module:Star_Formation_Feedback_Spheroids_Power_Law
File: tfamily star_formation.feedback.spheroids.power_law.F90
Default value: 3.5
Description: The velocity scaling of the SNe-driven outflow rate in spheroids.

Name: tfamily spheroidOutflowTimescaleMinimum
Attached to: tfamily module:Node_Component_Spheroid_Standard
File: tfamily objects.nodes.components.spheroid.standard.F90
Default value: $10^{-3}$
**Description:** The minimum timescale (in units of the spheroid dynamical time) on which outflows may deplete gas in the spheroid.

**Name:** tfamily spheroidOutflowVelocity  
**Attached to:** tfamily module:Star_Formation_Feedback_Spheroids_Power_Law  
**File:** tfamily star_formation.feedback.spheroids.power_law.F90  
**Default value:** 100 km s$^{-1}$  
**Description:** The velocity scale at which the SNe-driven outflow rate equals the star formation rate in spheroids.

**Name:** tfamily spheroidOutputStarFormationRate  
**Attached to:** tfamily module:Galacticus_Nodes  
**File:** tfamily objects.nodes.F90  
**Default value:** false  
**Description:** Specifies whether the starFormationRate method of the standard implementation of the spheroid component class should be output.

**Name:** tfamily spheroidSersicIndex  
**Attached to:** tfamily module:Node_Component_Spheroid_Standard  
**File:** tfamily objects.nodes.components.spheroid.standard.F90  
**Default value:** 4  
**Description:** The Sérsic index to use for the spheroid component mass distribution.

**Name:** tfamily spheroidSuperwindMassLoading  
**Attached to:** tfamily module:Star_Formation_Expulsive_Feedback_Spheroids_Superwind  
**File:** tfamily star_formation.feedback.expulsion.spheroids.superwind.F90  
**Default value:** 2  
**Description:** The mass loading of the spheroid superwind.

**Name:** tfamily spheroidSuperwindVelocity  
**Attached to:** tfamily module:Star_Formation_Expulsive_Feedback_Spheroids_Superwind  
**File:** tfamily star_formation.feedback.expulsion.spheroids.superwind.F90  
**Default value:** 200 km/s  
**Description:** The velocity scale of the spheroid superwind.

**Name:** tfamily spinDistributionBett2007Alpha  
**Attached to:** tfamily module:Halo_Spin_Distributions_Bett2007  
**File:** tfamily dark_matter_halos.spins.distributions.Bett2007.F90  
**Default value:** 2.509 [Bett et al., 2007]  
**Description:** The dispersion in a lognormal halo spin distribution.

**Name:** tfamily spinDistributionBett2007Lambda0  
**Attached to:** tfamily module:Halo_Spin_Distributions_Bett2007  
**File:** tfamily dark_matter_halos.spins.distributions.Bett2007.F90  
**Default value:** 0.04326 [Bett et al., 2007]  
**Description:** The median in a lognormal halo spin distribution.

**Name:** tfamily stabilityThresholdGaseous  
**Attached to:** tfamily module:Galactic_Dynamics_Bar_Instabilities_ELN
4. Input Parameters

File: tfamily galactic_dynamics.bar_instability.ELN.F90
Default value: 0.7
Description: The stability threshold in the Efstathiou et al. [1982] algorithm for purely gaseous disks.

Name: tfamily stabilityThresholdStellar
Attached to: tfamily module:Galactic_Dynamics_Bar_Instabilities_ELN
File: tfamily galactic_dynamics.bar_instability.ELN.F90
Default value: 1.1
Description: The stability threshold in the Efstathiou et al. [1982] algorithm for purely stellar disks.

Name: tfamily starFormationDiskEfficiency
Attached to: tfamily module:Star_Formation_Timescale_Disks_Dynamical_Time
File: tfamily star Formation.timescales.disks.dynamical_time.F90
Default value: 0.01
Description: The efficiency of star formation in disks for the dynamical time method.

Name: tfamily starFormationDiskMinimumTimescale
Attached to: tfamily module:Star_Formation_Timescale_Disks_Dynamical_Time
File: tfamily star Formation.timescales.disks.dynamical_time.F90
Default value: $10^{-3}$ Gyr
Description: The minimum timescale for star formation in disks.

Name: tfamily starFormationDiskTimescale
Attached to: unknown
Default value: 8.0
Description: The timescale (in Gyr) for star formation in the Baugh et al. [2005] prescription.

Name: tfamily starFormationDiskVelocityExponent
Attached to: tfamily module:Star_Formation_Timescale_Disks_Dynamical_Time
File: tfamily star Formation.timescales.disks.dynamical_time.F90
Default value: $-1.5$
Description: The velocity exponent for star formation in disks for the dynamical time method.

Name: tfamily starFormationExpansionExponent
Attached to: unknown
Default value: 0.0
Description: The exponent for expansion factor in the Baugh et al. [2005] prescription for star formation in galactic disks.

Name: tfamily starFormationExpulsiveFeedbackDisksMethod
Attached to: tfamily module:Star_Formation_Feedback_Expulsion_Disks
File: tfamily star Formation.feedback_expulsion.disks.F90
Default value: null
Description: The name of the method to be used for calculations of expulsive SNe feedback in disks.

Name: tfamily starFormationExpulsiveFeedbackSpheroidsMethod
Attached to: tfamily module:Star_Formation_Feedback_Expulsion_Spheroids
File: tfamily star_formation.feedback.expulsion.spheroids.F90
Default value: null
Description: The name of the method to be used for calculations of expulsive SNe feedback in spheroids.

Name: tfamily starFormationExtendedSchmidtGasExponent
Attached to: tfamily module:Star_Formation_Rate_Surface_Density_Disks_ExSchmidt
File: tfamily star_formation.rate_surface_density.disks.extended_Schmidt.F90
Default value: 1.0 [Shi et al., 2011]
Description: The exponent of gas surface density in the extended Schmidt star formation law.

Name: tfamily starFormationExtendedSchmidtNormalization
Attached to: tfamily module:Star_Formation_Rate_Surface_Density_Disks_ExSchmidt
File: tfamily star_formation.rate_surface_density.disks.extended_Schmidt.F90
Default value: $10^{-10.28}$ [Shi et al., 2011]
Description: The normalization of the extended Schmidt star formation law $M_\odot \text{yr}^{-1}\text{pc}^{-2}$.

Name: tfamily starFormationExtendedSchmidtStarExponent
Attached to: tfamily module:Star_Formation_Rate_Surface_Density_Disks_ExSchmidt
File: tfamily star_formation.rate_surface_density.disks.extended_Schmidt.F90
Default value: 0.48 [Shi et al., 2011]
Description: The exponent of stellar surface density in the extended Schmidt star formation law.

Name: tfamily starFormationFeedbackDisksCreasy2012Beta0
Attached to: tfamily module:Star_Formation_Feedback_Disks_Creasey2012
File: tfamily star_formation.feedback.disks.Creasey2012.F90
Default value: 13
Description: The factor $\beta_0$ appearing in the Creasey et al. [2012] model for supernovae feedback.

Name: tfamily starFormationFeedbackDisksCreasy2012Mu
Attached to: tfamily module:Star_Formation_Feedback_Disks_Creasey2012
File: tfamily star_formation.feedback.disks.Creasey2012.F90
Default value: 1.15
Description: The factor $\mu$ appearing in the Creasey et al. [2012] model for supernovae feedback.

Name: tfamily starFormationFeedbackDisksCreasy2012Nu
Attached to: tfamily module:Star_Formation_Feedback_Disks_Creasey2012
File: tfamily star_formation.feedback.disks.Creasey2012.F90
Default value: 0.16
Description: The factor $\nu$ appearing in the Creasey et al. [2012] model for supernovae feedback.

Name: tfamily starFormationFeedbackDisksMethod
Attached to: tfamily module:Star_Formation_Feedback_Disks
File: tfamily star_formation.feedback.disks.F90
Default value: powerLaw
Description: The name of the method to be used for calculations of SNe feedback in disks.

Name: tfamily starFormationFeedbackSpheroidsMethod
Attached to: tfamily module:Star_Formation_Feedback_Spheroids
File: tfamily star_formation.feedback.spheroids.F90
4. Input Parameters

- **Name:** tfamily starFormationFrequencyKMT09  
  **Attached to:** tfamily module:Star_Formation_Rate_Surface_Density_Disks_KMT09  
  **File:** tfamily star_formation.rate_surface_density.disks.KMT09.F90  
  **Default value:** 0.385 [Krumholz et al., 2009]  
  **Description:** The star formation frequency (in units of Gyr\(^{-1}\)) in the “Krumholz-McKee-Tumlinson” star formation timescale calculation.

- **Name:** tfamily starFormationFrequencyNormalizationBlitzRosolowsky  
  **Attached to:** tfamily module:Star_Formation_Rate_Surface_Density_Disks_BR  
  **File:** tfamily star_formation.rate_surface_density.disks.Blitz-Rosolowsky.F90  
  **Default value:** \(5.25 \times 10^{-10}\) [Leroy et al., 2008]  
  **Description:** The star formation frequency (in the low-density limit and in units of yr\(^{-1}\)) in the “Blitz-Rosolowsky” star formation timescale calculation.

- **Name:** tfamily starFormationHistoriesMethod  
  **Attached to:** tfamily module:Galacticus_Output_Star_Formation_Histories  
  **File:** tfamily galacticus.output.merger_tree.star_formation.F90  
  **Default value:** null  
  **Description:** The method to use for computing and outputting star formation histories.

- **Name:** tfamily starFormationHistoryFineTime  
  **Attached to:** tfamily module:Star_Formation_Histories_Metallicity_Split  
  **File:** tfamily galacticus.output.merger_tree.star_formation.metallicity_split.F90  
  **Default value:** 0.1  
  **Description:** The period prior to each output for which the fine time step is used in tabulations of star formation histories [Gyr].

- **Name:** tfamily starFormationHistoryFineTimeStep  
  **Attached to:** tfamily module:Star_Formation_Histories_Metallicity_Split  
  **File:** tfamily galacticus.output.merger_tree.star_formation.metallicity_split.F90  
  **Default value:** 0.01  
  **Description:** The fine time step to use in tabulations of star formation histories [Gyr].

- **Name:** tfamily starFormationHistoryMetallicityCount  
  **Attached to:** tfamily module:Star_Formation_Histories_Metallicity_Split  
  **File:** tfamily galacticus.output.merger_tree.star_formation.metallicity_split.F90  
  **Default value:** 10  
  **Description:** The number of bins in metallicity to use when tabulating star formation histories.

- **Name:** tfamily starFormationHistoryMetallicityMaximum  
  **Attached to:** tfamily module:Star_Formation_Histories_Metallicity_Split  
  **File:** tfamily galacticus.output.merger_tree.star_formation.metallicity_split.F90  
  **Default value:** 10\(^1\)  
  **Description:** The upper limit to the metallicity in the highest metallicity bin when tabulating star formation histories [Solar units].
Name: tfamily starFormationHistoryMetallicityMinimum
Attached to: tfamily module:Star_Formation_Histories_Metallicity_Split
File: tfamily galacticus.output.merger_tree.star_formation.metallicity_split.F90
Default value: $10^{-4}$
Description: The upper limit to the metallicity in the lowest metallicity bin when tabulating star formation histories [Solar units].

Name: tfamily starFormationHistoryTimeStep
Attached to: tfamily module:Star_Formation_Histories_Metallicity_Split
File: tfamily galacticus.output.merger_tree.star_formation.metallicity_split.F90
Default value: 0.1
Description: The time step to use in tabulations of star formation histories [Gyr].

Name: tfamily starFormationImfInstantaneousApproximation
Attached to: tfamily module:Star_Formation_IMF
File: tfamily star_formation.IMF.F90
Default value: false
Description: Option controlling whether stellar evolution should follow the instantaneous approximation.

Name: tfamily starFormationImfInstantaneousApproximationEffectiveAge
Attached to: tfamily module:Star_Formation_IMF
File: tfamily star_formation.IMF.F90
Default value: 13.8Gyr
Description: The effective age to use for computing SNeIa yield when using the instantaneous stellar evolution approximation.

Name: tfamily starFormationImfInstantaneousApproximationMassLongLived
Attached to: tfamily module:Star_Formation_IMF
File: tfamily star_formation.IMF.F90
Default value: $1M_{\odot}$
Description: The mass below which stars are assumed to be infinitely long-lived in the instantaneous approximation for stellar evolution.

Name: tfamily starFormationKennicuttSchmidtExponent
Attached to: tfamily module:Star_Formation_Rate_Surface_Density_Disks_KS
File: tfamily star_formation.rate_surface_density.disks.Kennicutt-Schmidt.F90
Default value: 1.4 [Kennicutt, 1998]
Description: The exponent in the Kennicutt-Schmidt star formation law.

Name: tfamily starFormationKennicuttSchmidtExponentTruncated
Attached to: tfamily module:Star_Formation_Rate_Surface_Density_Disks_KS
File: tfamily star_formation.rate_surface_density.disks.Kennicutt-Schmidt.F90
Default value: true
Description: The exponent of the $\Sigma_{gas}/\Sigma_{crit}$ term used in truncating the Kennicutt-Schmidt star formation law.

Name: tfamily starFormationKennicuttSchmidtNormalization
Attached to: tfamily module:Star_Formation_Rate_Surface_Density_Disks_KS

117
4. Input Parameters

File: tfamily star_formation.rate_surface_density.disks.Kennicutt-Schmidt.F90
Default value: 0.147 [Kennicutt, 1998]
Description: The normalization of the Kennicutt-Schmidt star formation law \([M_\odot \text{ Gyr}^{-1}\text{pc}^{-2}]\).

Name: tfamily starFormationKennicuttSchmidtTruncate
Attached to: tfamily module:Star_Formation_Rate_Surface_Density_Disks_KS
File: tfamily star_formation.rate_surface_density.disks.Kennicutt-Schmidt.F90
Default value: true
Description: Specifies whether or not to truncate star formation below a critical surface density in disks.

Name: tfamily starFormationRateSurfaceDensityDisksMethod
Attached to: tfamily module:Star_Formation_Rate_Surface_Density_Disks
File: tfamily star_formation.rate_surface_density.disks.F90
Default value: KMT09
Description: The name of the method to be used for computing star formation timescales in disks.

Name: tfamily starFormationSpheroidEfficiency
Attached to: tfamily module:Star_Formation_Timescale_Spheroids_Dynamical_Time
File: tfamily star_formation.timescales.spheroids.dynamical_time.F90
Default value: 0.04
Description: The efficiency of star formation in spheroids for the dynamical time method.

Name: tfamily starFormationSpheroidMinimumTimescale
Attached to: tfamily module:Star_Formation_Timescale_Spheroids_Dynamical_Time
File: tfamily star_formation.timescales.spheroids.dynamical_time.F90
Default value: 10^{-3} \text{ Gyr}
Description: The minimum timescale for star formation in disks.

Name: tfamily starFormationSpheroidVelocityExponent
Attached to: tfamily module:Star_Formation_Timescale_Spheroids_Dynamical_Time
File: tfamily star_formation.timescales.spheroids.dynamical_time.F90
Default value: 2.0
Description: The velocity exponent for star formation in spheroids for the dynamical time method.

Name: tfamily starFormationTimescaleDisksFixedTimescale
Attached to: tfamily module:Star_Formation_Timescale_Disks_Fixed
File: tfamily star_formation.timescales.disks.fixed.F90
Default value: 1 \text{ Gyr}
Description: The timescale for star formation in the fixed timescale model for disks.

Name: tfamily starFormationTimescaleDisksHaloScalingRedshiftExponent
Attached to: tfamily module:Star_Formation_Timescale_Disks_Halo_Scaling
File: tfamily star_formation.timescales.disks.halo_scaling.F90
Default value: 0
Description: The exponent of redshift in the timescale for star formation in the halo scaling timescale model for disks.

Name: tfamily starFormationTimescaleDisksHaloScalingTimescale
Attached to: tfamily module:Star_Formation_Timescale_Disks_Halo_Scaling
File: tfamily star_formation.timescales.disks.halo_scaling.F90
Default value: 1 Gyr
Description: The timescale for star formation in the halo scaling timescale model for disks.

Name: tfamily starFormationTimescaleDisksHaloScalingVirialVelocityExponent
Attached to: tfamily module:Star_Formation_Timescale_Disks_Halo_Scaling
File: tfamily star_formation.timescales.disks.halo_scaling.F90
Default value: 0
Description: The exponent of virial velocity in the timescale for star formation in the halo scaling timescale model for disks.

Name: tfamily starFormationTimescaleDisksMethod
Attached to: tfamily module:Star_Formation_Timescales_Disks
File: tfamily star_formation.timescales.disks.F90
Default value: integratedSurfaceDensity
Description: The name of the method to be used for computing star formation timescales in disks.

Name: tfamily starFormationTimescaleSpheroidsMethod
Attached to: tfamily module:Star_Formation_Timescales_Spheroids
File: tfamily star_formation.timescales.spheroids.F90
Default value: dynamicalTime
Description: The name of the method to be used for computing star formation timescales in spheroids.

Name: tfamily starveSatellites
Attached to: tfamily module:Node_Component_Hot_Halo_Standard
File: tfamily objects.nodes.components.hot_halo.standard.F90
Default value: false
Description: Specifies whether or not the hot halo should be removed ("starved") when a node becomes a satellite.

Name: tfamily stateFileRoot
Attached to: tfamily module:Galacticus_State
File: tfamily galacticus.state.F90
Default value: none
Description: The root name of files to which the internal state is written (to permit restarts).

Name: tfamily stateRetrieveFileRoot
Attached to: tfamily module:Galacticus_State
File: tfamily galacticus.state.F90
Default value: none
Description: The root name of files to which the internal state is retrieved from (to restart).

Name: tfamily stellarAstrophysicsMethod
Attached to: tfamily module:Stellar_Astrophysics
File: tfamily stellar_astrophysics.F90
Default value: file
Description: The name of the method to be used for stellar astrophysics calculations.

Name: tfamily stellarDensityChangeBinaryMotion
4. Input Parameters

**Attached to:** tfamily module:Black_Hole_Binary_Separations_Standard  
**File:** tfamily black_holes.binaries.separation_growth_rate.standard.F90  
**Default value:** true  
**Description:** The change in density due to the black hole’s motion.

**Name:** tfamily stellarFeedbackMethod  
**Attached to:** tfamily module:Stellar_Feedback  
**File:** tfamily stellar_astrophysics.feedback.F90  
**Default value:** standard  
**Description:** The method to use for computing aspects of stellar feedback.

**Name:** tfamily stellarMassFilterThreshold  
**Attached to:** tfamily module:Galacticus_Merger_Tree_Output_Filter_Stellar_Masses  
**File:** tfamily galacticus.output.merger_tree.filters.stellar_mass.F90  
**Default value:** none  
**Description:** The minimum stellar mass of a galaxy to pass the **stellarMass** output filter (in units of $M_\odot$).

**Name:** tfamily stellarMassMorphologyFilterRatioMaximum  
**Attached to:** tfamily module:Galacticus_Merger_Tree_Output_Stllr_Mss_Mrphlgs  
**File:** tfamily galacticus.output.merger_tree.filters.stellar_mass_morphology.F90  
**Default value:** none  
**Description:** The minimum spheroid-to-total ratio in stellar mass for which to allow output when the “stellarMassMorphology” output filter is active.

**Name:** tfamily stellarMassMorphologyFilterRatioMinimum  
**Attached to:** tfamily module:Galacticus_Merger_Tree_Output_Stllr_Mss_Mrphlgs  
**File:** tfamily galacticus.output.merger_tree.filters.stellar_mass_morphology.F90  
**Default value:** none  
**Description:** The minimum spheroid-to-total ratio in stellar mass for which to allow output when the “stellarMassMorphology” output filter is active.

**Name:** tfamily stellarPopulationLuminosityIntegrationToleranceRelative  
**Attached to:** tfamily module:Stellar_Population_Luminosities  
**File:** tfamily stellar_populations.luminosities.F90  
**Default value:** $10^{-3}$  
**Description:** The relative tolerance used when integrating the flux of stellar populations through filters.

**Name:** tfamily stellarPopulationLuminosityStoreToFile  
**Attached to:** tfamily module:Stellar_Population_Luminosities  
**File:** tfamily stellar_populations.luminosities.F90  
**Default value:** true  
**Description:** Specifies whether or not stellar populations luminosities (integrated under a filter) should be stored to file for rapid reuse.

**Name:** tfamily stellarPopulationPropertiesMethod  
**Attached to:** tfamily module:Stellar_Population_Properties  
**File:** tfamily stellar_populations.properties.F90  
**Default value:** instantaneous
Description: The method to use for computing properties of stellar populations.

Name: tfamily stellarPopulationSpectraFileForceZeroMetallicity
Attached to: tfamily module:Stellar_Population_Spectra_File
File: tfamily stellar_populations.spectra.file.F90
Default value: false
Description: Force the use of zero metallicity (or, lowest metallicity available) for all stellar populations.

Name: tfamily stellarPopulationSpectraMethod
Attached to: tfamily module:Stellar_Population_Spectra
File: tfamily stellar_populations.spectra.F90
Default value: Conroy-White-Gunn2009
Description: The name of the method to be used for calculations of stellar population spectra.

Name: tfamily stellarPopulationSpectraRecentTimeLimit
Attached to: tfamily module:Stellar_Population_Spectra_Postprocess
File: tfamily stellar_populations.spectra.postprocess.F90
Default value: 10⁷ years
Description: The maximum age of stellar populations to retain in the “recent” spectra postprocessing method.

Name: tfamily stellarPropertiesFile
Attached to: tfamily module:Stellar_Astrophysics_File
File: tfamily stellar_astrophysics.file.F90
Default value: data/stellarAstrophysics/Stellar_Properties_Compilation.xml
Description: The name of the XML file from which to read stellar properties (ejected masses, yields, etc.).

Name: tfamily stellarTracksFile
Attached to: tfamily module:Stellar_Astrophysics_Tracks_File
File: tfamily stellar_astrophysics.tracks.file.F90
Default value: data/stellarAstrophysics/Stellar_Tracks_Padova.hdf5
Description: The name of the HDF5 file from which to read stellar tracks.

Name: tfamily stellarTracksMethod
Attached to: tfamily module:Stellar_Astrophysics_Tracks
File: tfamily stellar_astrophysics.tracks.F90
Default value: file
Description: The name of the method to be used for stellar tracks calculations.

Name: tfamily stellarWindsMethod
Attached to: tfamily module:Stellar_Astrophysics_Winds
File: tfamily stellar_astrophysics.winds.F90
Default value: standard
Description: The method to use for computing aspects of stellar winds.

Name: tfamily summedNeutrinoMasses
Attached to: tfamily module:Transfer_Function_Eisenstein_Hu
File: tfamily structure_formation.transfer_function.Eisenstein_Hu.F90
4. Input Parameters

Default value: 0
Description: The summed mass (in electron volts) of all neutrino species.

Name: tfamily supernovaEnergy
Attached to: tfamily module:Stellar_Feedback_Standard
File: tfamily stellar_physics.feedback.standard.F90
Default value: $10^{51}$ ergs
Description: The energy produced by a supernova (in ergs).

Name: tfamily supernovaeIaMethod
Attached to: tfamily module:Supernovae_Type_Ia
File: tfamily stellar_physics.supernovae_type_Ia.F90
Default value: Nagashima
Description: The method to use for computing properties of Type Ia supernovae.

Name: tfamily supernovaePopIIIMethod
Attached to: tfamily module:Supernovae_Population_III
File: tfamily stellar_physics.supernovae_PopulationIII.F90
Default value: Heger-Woosley2002
Description: The method to use for computing properties of Population III supernovae.

Name: tfamily surfaceDensityCriticalBlitzRosolowsky
Attached to: tfamily module:Star_Formation_Rate_Surface_Density_Disks_BR
File: tfamily star_formation.rate_surface_density.disks.Blitz-Rosolowsky.F90
Default value: 200 [Bigiel et al., 2008]
Description: The surface density (in units of $M_\odot$ pc$^{-2}$) in the “Blitz-Rosolowsky” star formation timescale calculation at which low-density truncation begins.

Name: tfamily surfaceDensityExponentBlitzRosolowsky
Attached to: tfamily module:Star_Formation_Rate_Surface_Density_Disks_BR
File: tfamily star_formation.rate_surface_density.disks.Blitz-Rosolowsky.F90
Default value: 0.4 [Bigiel et al., 2008]
Description: The exponent for surface density in the “Blitz-Rosolowsky” star formation timescale calculation at in the high density regime.

Name: tfamily surveyGeometryMethod
Attached to: tfamily module:Geometry_Surveys
File: tfamily geometry.surveys.F90
Default value: none
Description: The name of the method to be used for computing survey geometries.

Name: tfamily tidalStrippingMassLossRateDiskSimpleFractionalRateMaximum
Attached to: tfamily module:Tidal_Stripping_Mass_Loss_Rate_Disks_Simple
File: tfamily tidal_stripping.mass_loss_rate.disks.simple.F90
Default value: 10
Description: The maximum fractional mass loss rate per dynamical time in the simple model of mass loss from disks due to tidal stripping.

Name: tfamily tidalStrippingMassLossRateDisksMethod
Attached to: tfamily module:Tidal_Stripping_Mass_Loss_Rate_Disks
File: tfamily tidal_stripping.mass_loss_rate.disks.F90
Default value: null
Description: The name of the method to be used when computing mass loss rates from disks due to tidal stripping.

Name: tfamily tidalStrippingMassLossRateSpheroidSimpleFractionalRateMaximum
Attached to: tfamily module:Tidal_Stripping_Mass_Loss_Rate_Spheroids_Simple
File: tfamily tidal_stripping.mass_loss_rate.spheroids.simple.F90
Default value: 10
Description: The maximum fractional mass loss rate per dynamical time in the simple model of mass loss from spheroids due to tidal stripping.

Name: tfamily tidalStrippingMassLossRateSpheroidsMethod
Attached to: tfamily module:Tidal_Stripping_Mass_Loss_Rate_Spheroids
File: tfamily tidal_stripping.mass_loss_rate.spheroids.F90
Default value: null
Description: The name of the method to be used when computing mass loss rates from spheroids due to tidal stripping.

Name: tfamily timePerTreeFitFileName
Attached to: tfamily module:Galacticus_Meta_Compute_Times_File
File: tfamily galacticus.meta.compute_times.file.F90
Default value: none
Description: The name of the file which contains fit coefficients for the time per tree fitting function.

Name: tfamily timePerTreeMethod
Attached to: tfamily module:Galacticus_Meta_Compute_Times
File: tfamily galacticus.meta.compute_times.F90
Default value: file
Description: The name of the method to be used for computing the time per tree.

Name: tfamily timestepHistoryBegin
Attached to: tfamily module:Merger_Tree_Timesteps_History
File: tfamily merger_trees.evolve.timesteps.history.F90
Default value: 5% of the age of the Universe
Description: The earliest time at which to tabulate the volume averaged history of galaxies (in Gyr).

Name: tfamily timestepHistoryEnd
Attached to: tfamily module:Merger_Tree_Timesteps_History
File: tfamily merger_trees.evolve.timesteps.history.F90
Default value: The age of the Universe
Description: The latest time at which to tabulate the volume averaged history of galaxies (in Gyr).

Name: tfamily timestepHistorySteps
Attached to: tfamily module:Merger_Tree_Timesteps_History
File: tfamily merger_trees.evolve.timesteps.history.F90
Default value: 30
Description: The number of steps (spaced logarithmically in cosmic time) at which to tabulate the
4. Input Parameters

volume averaged history of galaxies.

**Name**: tfamily timestepHostAbsolute  
**Attached to**: tfamily module:Merger_Trees_Evolve  
**File**: tfamily merger_trees.evolve.F90  
**Default value**: 1  
**Description**: The maximum allowed absolute timestep (in Gyr) for node evolution relative to the time of the host halo.

**Name**: tfamily timestepHostRelative  
**Attached to**: tfamily module:Merger_Trees_Evolve  
**File**: tfamily merger_trees.evolve.F90  
**Default value**: 0.1  
**Description**: The maximum allowed relative timestep for node evolution relative to the time of the host halo.

**Name**: tfamily timestepRecordEvolution  
**Attached to**: tfamily module:Merger_Tree_Timesteps_Record_Evolution  
**File**: tfamily merger_trees.evolve.timesteps.record_evolution.F90  
**Default value**: false  
**Description**: Specifies whether or not the evolution of the main branch galaxy should be recorded.

**Name**: tfamily timestepRecordEvolutionBegin  
**Attached to**: tfamily module:Merger_Tree_Timesteps_Record_Evolution  
**File**: tfamily merger_trees.evolve.timesteps.record_evolution.F90  
**Default value**: 5% of the age of the Universe  
**Description**: The earliest time at which to tabulate the evolution of main branch progenitor galaxies (in Gyr).

**Name**: tfamily timestepRecordEvolutionEnd  
**Attached to**: tfamily module:Merger_Tree_Timesteps_Record_Evolution  
**File**: tfamily merger_trees.evolve.timesteps.record_evolution.F90  
**Default value**: The age of the Universe  
**Description**: The latest time at which to tabulate the evolution of main branch progenitor galaxies (in Gyr).

**Name**: tfamily timestepRecordEvolutionSteps  
**Attached to**: tfamily module:Merger_Tree_Timesteps_Record_Evolution  
**File**: tfamily merger_trees.evolve.timesteps.record_evolution.F90  
**Default value**: 30  
**Description**: The number of steps (spaced logarithmically in cosmic time) at which to tabulate the evolution of main branch progenitor galaxies.

**Name**: tfamily timestepSimpleAbsolute  
**Attached to**: tfamily module:Merger_Tree_Timesteps_Simple  
**File**: tfamily merger_trees.evolve.timesteps.simple.F90  
**Default value**: 1  
**Description**: The maximum allowed absolute change in time (in Gyr) for a single step in the evolution of a node.
Name: tfamily timestepSimpleRelative  
**Attached to:** tfamily module:Merger_Tree_Timesteps_Simple  
**File:** tfamily merger_trees.evolve.timesteps.simple.F90  
**Default value:** 0.1  
**Description:** The maximum allowed relative change in time for a single step in the evolution of a node.

Name: tfamily toomreParameterCritical  
**Attached to:** tfamily module:Star_Formation_Rate_Surface_Density_Disks_KS  
**File:** tfamily star_formation.rate_surface_density.disks.Kennicutt-Schmidt.F90  
**Default value:** 0.4 [Kennicutt, 1989]  
**Description:** The critical Toomre parameter for star formation in disks.

Name: tfamily transferFunction  
**Attached to:** tfamily module:Merger_Trees_Simple  
**File:** tfamily merger_trees.file_maker.simple.F90  
**Default value:** none  
**Description:** The type of transfer function used.

Name: tfamily transferFunctionFile  
**Attached to:** tfamily module:Transfer_Functions_File  
**File:** tfamily structure_formation.transfer_function.file.F90  
**Default value:** none  
**Description:** The name of a file containing a tabulation of the transfer function for the “file” transfer function method.

Name: tfamily transferFunctionMethod  
**Attached to:** tfamily module:Transfer_Functions  
**File:** tfamily structure_formation.transfer_function.F90  
**Default value:** Eisenstein-Hu1999  
**Description:** The name of the method to be used for computing the transfer function.

Name: tfamily transferFunctionWDMFreeStreamingLength  
**Attached to:** tfamily module:Transfer_Function_BBKS  
**File:** tfamily structure_formation.transfer_function.BBKS.F90  
**Default value:** 0  
**Description:** The warm dark matter free streaming length (in Mpc).

Name: tfamily transferFunctionWdmCutOffScale  
**Attached to:** tfamily module:Transfer_Function_Eisenstein_Hu  
**File:** tfamily structure_formation.transfer_function.Eisenstein_Hu.F90  
**Default value:** 0  
**Description:** The cut-off scale in the transfer function due to warm dark matter.

Name: tfamily transferFunctionWdmEpsilon  
**Attached to:** tfamily module:Transfer_Function_Eisenstein_Hu  
**File:** tfamily structure_formation.transfer_function.Eisenstein_Hu.F90  
**Default value:** 0.361 [Barkana et al., 2001]  
**Description:** The parameter $\epsilon$ appearing in the warm dark matter transfer function [Barkana et al.,...
4. Input Parameters

Name: tfamily transferFunctionWdmEta
Attached to: tfamily module:Transfer_Function_Eisenstein_Hu
File: tfamily structure_formation.transfer_function.Eisenstein_Hu.F90
Default value: 5.0 [Barkana et al., 2001]
Description: The parameter $\epsilon$ appearing in the warm dark matter transfer function [Barkana et al., 2001].

Name: tfamily transferFunctionWdmNu
Attached to: tfamily module:Transfer_Function_Eisenstein_Hu
File: tfamily structure_formation.transfer_function.Eisenstein_Hu.F90
Default value: 1.2 [Barkana et al., 2001]
Description: The parameter $\epsilon$ appearing in the warm dark matter transfer function [Barkana et al., 2001].

Name: tfamily treeBranchingMethod
Attached to: tfamily module:Merger_Tree_Branching
File: tfamily merger_trees.branching_probability.F90
Default value: modifiedPress-Schechter
Description: The name of the method to be used for computing merger tree branching probabilities when building merger trees.

Name: tfamily treeBranchingModifierMethod
Attached to: tfamily module:Merger_Tree_Branching_Modifiers
File: tfamily merger_trees.branching_probability.modifier.F90
Default value: null
Description: The name of the method to be used for computing modifiers to merger tree branching probabilities.

Name: tfamily treeEvolveLimitLoadAverage
Attached to: tfamily module:Galacticus_Tasks_Evolve_Tree
File: tfamily galacticus.tasks.evolve_tree.F90
Default value: false
Description: Specifies whether or not to limit the load average

Name: tfamily treeEvolveLoadAverageMaximum
Attached to: tfamily module:Galacticus_Tasks_Evolve_Tree
File: tfamily galacticus.tasks.evolve_tree.F90
Default value: processorCount
Description: The maximum load average for which new trees will be processed.

Name: tfamily treeEvolveThreadLock
Attached to: tfamily module:Galacticus_Tasks_Evolve_Tree
File: tfamily galacticus.tasks.evolve_tree.F90
Default value: false
Description: Specifies whether or not to limit the number of threads across all GALACTICUS processes.

Name: tfamily treeEvolveThreadsMaximum
Attached to: tfamily module:Galacticus_Tasks_Evolve_Tree
File: tfamily galacticus.tasks.evolve_tree.F90
Default value: processorCount
Description: The maximum number of active threads across all GALACTICUS processes.

Name: tfamily treeEvolveWorkerCount
Attached to: tfamily module:Galacticus_Tasks_Evolve_Tree
File: tfamily galacticus.tasks.evolve_tree.F90
Default value: 1
Description: The number of workers that will work on this calculation.

Name: tfamily treeEvolveWorkerNumber
Attached to: tfamily module:Galacticus_Tasks_Evolve_Tree
File: tfamily galacticus.tasks.evolve_tree.F90
Default value: 1
Description: The number of this worker.

Name: tfamily treeNodeMethodAgeStatistics
Attached to: tfamily module:Galacticus_Nodes
File: tfamily objects.nodes.F90
Default value: null
Description: Specifies the implementation to be used for the ageStatistics component of nodes.

Name: tfamily treeNodeMethodBasic
Attached to: tfamily module:Galacticus_Nodes
File: tfamily objects.nodes.F90
Default value: standard
Description: Specifies the implementation to be used for the basic component of nodes.

Name: tfamily treeNodeMethodBlackHole
Attached to: tfamily module:Galacticus_Nodes
File: tfamily objects.nodes.F90
Default value: standard
Description: Specifies the implementation to be used for the blackHole component of nodes.

Name: tfamily treeNodeMethodDarkMatterProfile
Attached to: tfamily module:Galacticus_Nodes
File: tfamily objects.nodes.F90
Default value: scale
Description: Specifies the implementation to be used for the darkMatterProfile component of nodes.

Name: tfamily treeNodeMethodDisk
Attached to: tfamily module:Galacticus_Nodes
File: tfamily objects.nodes.F90
Default value: exponential
Description: Specifies the implementation to be used for the disk component of nodes.

Name: tfamily treeNodeMethodDynamicsStatistics
Attached to: tfamily module:Galacticus_Nodes
127
4. Input Parameters

File: tfamily objects.nodes.F90
Default value: null
Description: Specifies the implementation to be used for the dynamicsStatistics component of nodes.

Name: tfamily treeNodeMethodFormationTime
Attached to: tfamily module:Galacticus_Nodes
File: tfamily objects.nodes.F90
Default value: null
Description: Specifies the implementation to be used for the formationTime component of nodes.

Name: tfamily treeNodeMethodHostHistory
Attached to: tfamily module:Galacticus_Nodes
File: tfamily objects.nodes.F90
Default value: null
Description: Specifies the implementation to be used for the hostHistory component of nodes.

Name: tfamily treeNodeMethodHotHalo
Attached to: tfamily module:Galacticus_Nodes
File: tfamily objects.nodes.F90
Default value: standard
Description: Specifies the implementation to be used for the hotHalo component of nodes.

Name: tfamily treeNodeMethodIndices
Attached to: tfamily module:Galacticus_Nodes
File: tfamily objects.nodes.F90
Default value: null
Description: Specifies the implementation to be used for the indices component of nodes.

Name: tfamily treeNodeMethodInterOutput
Attached to: tfamily module:Galacticus_Nodes
File: tfamily objects.nodes.F90
Default value: null
Description: Specifies the implementation to be used for the interOutput component of nodes.

Name: tfamily treeNodeMethodMassFlowStatistics
Attached to: tfamily module:Galacticus_Nodes
File: tfamily objects.nodes.F90
Default value: null
Description: Specifies the implementation to be used for the massFlowStatistics component of nodes.

Name: tfamily treeNodeMethodMergingStatistics
Attached to: tfamily module:Galacticus_Nodes
File: tfamily objects.nodes.F90
Default value: null
Description: Specifies the implementation to be used for the mergingStatistics component of nodes.

Name: tfamily treeNodeMethodNBody
Attached to: tfamily module:Galacticus_Nodes
File: tfamily objects.nodes.F90
**Default value:** null
**Description:** Specifies the implementation to be used for the nBody component of nodes.

**Name:** tfamily treeNodeMethodPosition
**Attached to:** tfamily module:Galacticus_Nodes
**File:** tfamily objects.nodes.F90
**Default value:** null
**Description:** Specifies the implementation to be used for the position component of nodes.

**Name:** tfamily treeNodeMethodSatellite
**Attached to:** tfamily module:Galacticus_Nodes
**File:** tfamily objects.nodes.F90
**Default value:** standard
**Description:** Specifies the implementation to be used for the satellite component of nodes.

**Name:** tfamily treeNodeMethodSpheroid
**Attached to:** tfamily module:Galacticus_Nodes
**File:** tfamily objects.nodes.F90
**Default value:** standard
**Description:** Specifies the implementation to be used for the spheroid component of nodes.

**Name:** tfamily treeNodeMethodSpin
**Attached to:** tfamily module:Galacticus_Nodes
**File:** tfamily objects.nodes.F90
**Default value:** random
**Description:** Specifies the implementation to be used for the spin component of nodes.

**Name:** tfamily tripleBlackHoleInteraction
**Attached to:** tfamily module:Node_Component_Black_Hole_Standard
**File:** tfamily objects.nodes.components.black_hole.standard.F90
**Default value:** false
**Description:** Determines whether or not triple black hole interactions will be accounted for.

**Name:** tfamily velocityDispersionDiskGas
**Attached to:** tfamily module:Star_Formation_Rate_Surface_Density_Disks_KS
**File:** tfamily star Formation.rate_surface_density.disks.Kennicutt-Schmidt.F90
**Default value:** 10 [Leroy et al., 2008]
**Description:** The velocity dispersion of gas in disks.

**Name:** tfamily verbosityLevel
**Attached to:** tfamily module:Halo_Mass_Function_Tasks
**File:** tfamily halo_mass_functions.tasks.F90
**Default value:** 1
**Description:** The level of verbosity for GALACTICUS (higher values give more verbosity).

**Name:** tfamily virialDensityContrastFixed
**Attached to:** tfamily module:Virial_Density_Contrast
**File:** tfamily structure_formation.virial_density_contrast.F90
**Default value:** 200
4. Input Parameters

**Description:** The virial density contrast to use in the fixed value model.

**Name:** tfamily virialDensityContrastFixedType  
**Attached to:** tfamily module:Virial_Density_Contrast  
**File:** tfamily structure_formation.virial_density_contrast.F90  
**Default value:** criticalDensity  
**Description:** The reference density to use in the fixed value virial density contrast model. Either of critical density and mean density are allowed.

**Name:** tfamily virialDensityContrastFoFDensityRatio  
**Attached to:** tfamily module:Virial_Density_Contrast  
**File:** tfamily structure_formation.virial_density_contrast.F90  
**Default value:** 4.688 (value appropriate for an Navarro-Frenk-White (dark matter halo profile) (NFW) profile with concentration $c = 6.88$ which is the concentration found by Prada et al. [2011] for halos with $\sigma = 1.686$ which is the approximate critical overdensity for collapse).  
**Description:** The ratio of mean virial density to density at the virial radius to assume when setting virial density contrasts in the friends-of-friends model.

**Name:** tfamily virialDensityContrastFoFLinkingLength  
**Attached to:** tfamily module:Virial_Density_Contrast  
**File:** tfamily structure_formation.virial_density_contrast.F90  
**Default value:** 0.2  
**Description:** The friends-of-friends linking length algorithm to use in computing virial density contrast.

**Name:** tfamily virialDensityContrastMethod  
**Attached to:** tfamily module:Virial_Density_Contrast  
**File:** tfamily structure_formation.virial_density_contrast.F90  
**Default value:** sphericalCollapseMatterLambda  
**Description:** The method to be used for virialDensityContrast.

**Name:** tfamily virialDensityContrastSphericalTopHatDarkEnergyFixEnergyAt  
**Attached to:** tfamily module:Spherical_Collapse_Matter_Dark_Energy  
**File:** tfamily structure_formation.spherical_collapse.matter_dark_energy.F90  
**Default value:** turnaround  
**Description:** Selects the epoch at which the energy of a spherical top hat perturbation in a dark energy cosmology should be “fixed” for the purposes of computing virial density contrasts. (See the discussion in Percival 2005; §8.)

**Name:** tfamily virialOrbitsFixedRadialVelocity  
**Attached to:** tfamily module:Virial_Orbits_Fixed  
**File:** tfamily satellites.merging.virial_orbits.fixed.F90  
**Default value:** 0.90  
**Description:** The radial velocity (in units of the host virial velocity) to used for the fixed virial orbits distribution. Default value matches approximate peak in the distribution of Benson [2005].

**Name:** tfamily virialOrbitsFixedTangentialVelocity  
**Attached to:** tfamily module:Virial_Orbits_Fixed  
**File:** tfamily satellites.merging.virial_orbits.fixed.F90  
**Default value:** 0.75
**Description:** The radial velocity (in units of the host virial velocity) to used for the fixed virial orbits distribution. Default value matches approximate peak in the distribution of Benson [2005].

**Name:** tfamily virialOrbitsMethod  
**Attached to:** tfamily module:Virial_Orbits  
**File:** tfamily satellites.merging.virial_orbits.F90  
**Default value:** Benson2005  
**Description:** Selects the method to be used for finding orbital parameters of satellites at virial radius crossing.

**Name:** tfamily warmDarkMatterCriticalOverdensityGX  
**Attached to:** tfamily module:Critical_Overdensity_Mass_Scalings_WDM  
**File:** tfamily structure_formation.critical_overdensity.mass_scaling.warm_dark_matter.F90  
**Default value:** 1.5  
**Description:** The effective number of degrees of freedom for the warm dark matter particle.

**Name:** tfamily warmDarkMatterCriticalOverdensityMX  
**Attached to:** tfamily module:Critical_Overdensity_Mass_Scalings_WDM  
**File:** tfamily structure_formation.critical_overdensity.mass_scaling.warm_dark_matter.F90  
**Default value:** 1.0 keV  
**Description:** The mass (in keV) of the warm dark matter particle.

**Name:** tfamily warmDarkMatterCriticalOverdensityUseFittingFunction  
**Attached to:** tfamily module:Critical_Overdensity_Mass_Scalings_WDM  
**File:** tfamily structure_formation.critical_overdensity.mass_scaling.warm_dark_matter.F90  
**Default value:** true  
**Description:** Specifies whether the warm dark matter critical overdensity mass scaling should be computed from a fitting function or from tabulated data.

**Name:** tfamily zeroCoolingRateAboveVelocity  
**Attached to:** tfamily module:Cooling_Rates_White-Frenk  
**File:** tfamily cooling.cooling_rate.White-Frenk.F90  
**Default value:** 10000  
**Description:** The halo virial velocity (in km/s) above which cooling rates are forced to zero in the White-Frenk1991 cooling rate model.
5. Extracting and Analyzing Results

Galacticus stores its output in an HDF5 file. The contents of this file can be viewed and manipulated using a variety of ways including:

**HDFView** This is a graphical viewer for exploring the contents of HDF5 files;

**HDF5 Command Line Tools** A set of tools which can be used to extract data from HDF5 files ([h5dump](#) and **h5ls** are particularly useful);

**C++ and Fortran 90 APIs** Allow access to and manipulation of data in HDF5 files;

**h5py** A Python interface to HDF5 files.

In the remainder of this section the structure of Galacticus HDF5 files is described and a general-purpose Perl module which we use to extract data in a convenient manner is outlined.

5.1. General Structure of Output File

Figure 5.1 shows the structure of a typical Galacticus output file. The various groups and subgroups are described below.

5.1.1. UUID

The UUID (**Universally Unique Identifier**) is a unique identifier assigned to each Galacticus model that is run. It allows identification of a given model and can be referenced from, for example, an external database. Using the `Galacticus::HDF5` Perl module (see §5.2), the UUID can be loaded into the data structure using:

```perl
&HDF5::Get_UUID($model);
```

The UUID is then available as `$model->'uuid'`.

5.1.2. Build Information

Galacticus automatically stores various information about how it was built in the **Build** group attributes. Currently, included attributes consist of:

- **FGSL_library_version** The version number of the FGSL library;
- **FoX_library_version** The version number of the FoX library;
- **GSL_library_version** The version number of the GSL library;
- **HDF5_library_version** The version number of the HDF5 library;
- **make_CCOMPILER** The C compiler command used;
- **make_CCOMPILER_VERSION** The C compiler version information;
Figure 5.1.: Structure of a Galacticus HDF5 output file. 

<treeCount> is the total number of merger trees present in a given output, and <nodeCount> is the total number of nodes (in all trees) present in an output.
5.1. General Structure of Output File

make_CFLAGS The flags passed to the C compiler;
make_CPPCOMPILER The C++ compiler command used;
make_CPPCOMPILER_VERSION The C++ compiler version information;
make_CPPFLAGS The flags passed to the C++ compiler;
make_FCCOMPILER The Fortran compiler command used;
make_FCCOMPILER_VERSION The Fortran compiler version information;
make_FCFLAGS The flags passed to the Fortran compiler;
make_FCFLAGS_NOOPT The flags passed to the Fortran compiler for unoptimized compiles;
make_MODULETYPE The Fortran module type identifier string;
make_PREPROCESSOR The preprocessor command used.

Additionally, two datasets are included which store details of the GALACTICUS source changeset. sourceChangeSetMerge contains the output of “hg bundle -t none”, that is, it contains a Mercurial changegroup that incorporates any changes made to the current branch relative to the main GALACTICUS branch. sourceChangeSetDiff contains the output of “hg diff”, that is, all differences between the source code in the working directory and that which has been committed to Mercurial. Used together, these two datasets allow the precise source code used to run the model to be recovered from the main branch GALACTICUS source.

5.1.3. Parameters

The Parameters group contains a record of all parameter values (either input or default) that were used for this GALACTICUS run. The group contains a long list of attributes, each attribute named for the corresponding parameter and with a single entry giving the value of that parameter. The scripts/aux/Extract_Parameter_File.pl script can be used to extract these parameter values to an XML file suitable for re-input into GALACTICUS.

5.1.4. Version

The Version group contains a record of the GALACTICUS version used for this model, storing the major and minor version numbers, the revision number and the MERCURIAL revision and hash (if the code is being maintained using MERCURIAL, otherwise a value of −1 is entered or the revision and the hash attribute is empty). Additionally, the time at which the model was run is stored and, if the galacticusConfig.xml file (see §3.1) is present and contains contact details, the name and e-mail address of the person who ran the model.

5.1.5. globalHistory

The globalHistory group stores volume averaged properties of the model universe as a function of time. Currently, the properties stored are:

historyTime Cosmic time (in Gyr);
historyExpansion Expansion factor;
historyStarFormationRate Volume averaged star formation rate (in $M_\odot$/Gyr/Mpc$^3$).
5. Extracting and Analyzing Results

- **historyDiskStarFormationRate**: Volume averaged star formation rate in disks (in \(M_\odot/\text{Gyr}/\text{Mpc}^3\)).
- **historySpheroidStarFormationRate**: Volume averaged star formation rate in spheroids (in \(M_\odot/\text{Gyr}/\text{Mpc}^3\)).
- **historyStellarDensity**: Volume averaged stellar mass density (in \(M_\odot/\text{Mpc}^3\)).
- **historyDiskStellarDensity**: Volume averaged stellar mass density in disks (in \(M_\odot/\text{Mpc}^3\)).
- **historySpheroidStellarDensity**: Volume averaged stellar mass density in spheroids (in \(M_\odot/\text{Mpc}^3\)).
- **historyGasDensity**: Volume averaged cooled gas density (in \(M_\odot/\text{Mpc}^3\)).
- **historyNodeDensity**: Volume averaged resolved node density (in \(M_\odot/\text{Mpc}^3\)).

Dimensional datasets have a `unitsInSI` attribute which gives their units in the SI system.

### 5.1.6. Outputs

The `Outputs` group contains one or more sub-groups corresponding to the output times requested from **GALACTICUS**. Each sub-group contains the following information:

- **outputTime** *(attribute)* The cosmic time (in Gyr) at this output;
- **outputExpansionFactor** *(attribute)* The expansion factor at this output;
- **nodeData** A group of node properties as described below.

#### nodeData group

The `nodeData` group contains all data from nodes in all merger trees. The group consists of a collection of datasets each of which lists a property of all nodes in the trees which exist at the output time. Where relevant, each dataset contains an attribute, `unitsInSI`, which gives the units of the dataset in the SI system.

#### mergerTree datasets

To allow locating of nodes belonging to a given merger tree in the datasets in the `nodeData` group, the `mergerTreeStartIndex` and `mergerTreeCount` datasets list the starting index of each tree’s nodes in the `nodeData` datasets, and the number of nodes belonging to each tree respectively. Additionally, the `mergerTreeWeight` dataset lists the `volumeWeight` property for each tree (see §5.1.6) which gives the weight (in \(\text{Mpc}^{-3}\)) which should be assigned to this tree (and all nodes in it) to create a volume-averaged sample (see §5.3.1). Finally, the `mergerTreeIndex` dataset gives the index of each tree stored in the `nodeData` datasets.

#### mergerTree subgroups

These subgroups will be present if the `[mergerTreeOutputReferences]` parameter is set to true. Each `mergerTree` subgroup contains HDF5 references to all data on a single merger tree. The group consists of a collection of scalar references each of which points to the appropriate region of the corresponding dataset in the `nodeData` group. Additionally, the `volumeWeight` attribute of this group gives the weight (in \(\text{Mpc}^{-3}\)) which should be assigned to this tree (and all nodes in it) to create a volume-averaged sample. (A second attribute, `volumeWeightUnitsInSI`, gives the units of `volumeWeight` in the SI system.)
5.1.7. Optional Outputs

Numerous other quantities can be optionally output. These are documented below:

**Redshifts**

The redshift corresponding to the time at which a node was last isolated can be output by setting \[\text{outputNodeRedshifts}\] to \textbf{true}. This quantity will be output as \textit{basicRedshiftLastIsolated}.

**Mass Accretion Histories**

A mass accretion history (i.e. mass as a function of time) for the main branch in each merger tree can be output by setting \textit{massAccretionHistoryOutput} = \textbf{true}. If requested, a new group \textit{massAccretionHistories} will be made in the GALACTICUS output file. It will contain groups called \textit{mergerTreeN} where \(N\) is the merger tree index. Each such group will contain the following three datasets, defined for the main branch of the tree\(^1\):

\begin{itemize}
  \item \texttt{nodeIndex} The index of the node in the tree;
  \item \texttt{nodeTime} The time at this point in the tree (in Gyr);
  \item \texttt{nodeMass} The mass of the node at this point in the tree (in \(M_\odot\)). The \texttt{nodeMass} property is defined to be the total mass of each node in a merger tree. Therefore, it includes both dark and baryonic mass. Additionally, the mass of a node includes the mass of any satellite nodes that it may contain. The mean density of the node depends on the method selected by the \texttt{virialDensityContrastMethod} parameter.
\end{itemize}

**Merger Tree Dump**

A full dump of merger tree structure by setting \texttt{mergerTreeStructureDump} = \textbf{true}. In this case, files will be dumped to the directory specified by \texttt{[mergerTreeStructureDumpDirectory]} for each merger tree with final mass between \texttt{[mergerTreeStructureDumpMassMinimum]} and \texttt{[mergerTreeStructureDumpMassMaximum]}. Each tree is dumped to a file named “\texttt{mergerTreeDump:<treeIndex>:1.gv}” in the specified directory in \textsc{GraphViz} format.

**Conditional Mass Functions**

Setting \texttt{[mergerTreeComputeConditionalMassFunction]} = \textbf{true} will cause conditional mass functions to be computed and output to the GALACTICUS output file in a group named “\texttt{conditionalMassFunction}”. The mass functions are binned in parent halo mass, and the mass ratio of the progenitor to parent halo. Bins are logarithmically spaced in mass (and mass ratio), with the range and number of bins controlled by the parameters:

\begin{itemize}
  \item \texttt{[mergerTreeComputeConditionalMassFunctionParentMassCount]};
  \item \texttt{[mergerTreeComputeConditionalMassFunctionParentMassMinimum]};
  \item \texttt{[mergerTreeComputeConditionalMassFunctionParentMassMaximum]};
  \item \texttt{[mergerTreeComputeConditionalMassFunctionMassRatioCount]};
  \item \texttt{[mergerTreeComputeConditionalMassFunctionMassRatioMinimum]};
\end{itemize}

\(^1\)“Main branch” is defined by starting from the root node of a tree and repeatedly stepping back to the most massive progenitor of the branch. This does not necessarily pick out the most massive progenitor at a given time.
5. Extracting and Analyzing Results

- \[\text{mergerTreeComputeConditionalMassFunctionMassRatioMaximum}\].

The resulting parent masses and mass ratios are written to datasets \text{massParent} and \text{massRatio} respectively. Parent and progenitor halos are defined at a set of redshifts defined by the arrays \[\text{mergerTreeComputeConditionalMassFunctionParentRedshifts}\] and \[\text{mergerTreeComputeConditionalMassFunctionProgenitorRedshifts}\], which are written to datasets \text{redshiftParent} and \text{redshiftProgenitor}. The resulting conditional masses functions are written to datasets \text{conditionalMassFunction} and \text{conditionalMassFunctionError}.

In addition to standard progenitor mass functions, the progenitor mass function conditioned on progenitor rank (i.e. \(1^{\text{st}}\) most massive, \(2^{\text{nd}}\), ..., \(n^{\text{th}}\) most massive progenitor) is computed and output to the datasets \text{primaryProgenitorMassFunction} and \text{primaryProgenitorMassFunctionError}. The depth (i.e. \(n\)) is specified by \[\text{mergerTreeComputeConditionalMassFunctionPrimaryProgenitorDepth}\].

Finally, the progenitor mass function conditioned on recent formation is computed and output to the datasets \text{formationRateFunction} and \text{formationRateFunctionError}. To be considered “recently formed” a progenitor must have formed between \(t\) and \(t(1 - \Delta)\) where \(t\) is the progenitor time and \(\Delta = \text{mergerTreeConditionalMassFunctionFormationRateTimeFraction}\).

Pre-Evolution Merger Trees

\text{GALACTICUS} can output the full structure of merger trees prior to any evolution. Merger tree structure can be requested by setting \text{mergerTreeStructureOutput=true}. Structures are written to a new group, \text{mergerTreeStructures}, in the \text{GALACTICUS} output file. This group will contain groups called \text{mergerTreeN} where \(N\) is the merger tree index. Each such group will contain the following datasets:

- \text{nodeIndex} The index of the node in the tree;
- \text{childIndex} The index of this node’s first child node;
- \text{parentIndex} The index of this node’s parent node;
- \text{SiblingIndex} The index of this node’s sibling node;
- \text{nodeTime} The time at this point in the tree (in Gyr);
- \text{nodeMass} The mass of the node at this point in the tree (in \(M_\odot\)). The \text{nodeMass} property is defined to be the total mass of each node in a merger tree. Therefore, it includes both dark and baryonic mass. Additionally, the mass of a node includes the mass of any satellite nodes that it may contain. The mean density of the node depends on the method selected by the \text{virialDensityContrastMethod} parameter.

Additional, optional, datasets can be added by setting appropriate input parameters. Currently these include:

- **Virial quantities** If \text{mergerTreeStructureOutputVirialQuantities=true} then two additional datasets are included:
  - \text{nodeVirialRadius} The virial radius of the node (in Mpc);
  - \text{nodeVirialVelocity} The virial velocity of the node (in km/s);
- **Dark matter scale radii** If \text{mergerTreeStructureOutputDarkMatterScaleRadius=true} then an additional dataset is included:
  - \text{darkMatterScaleRadius} The scale radius of this node’s dark matter halo profile (in Mpc);
- **Tree final descendent** If \text{outputFinalDescendentIndices=true} then an additional dataset is included:
  - \text{finalDescendentIndex} The index of the final descendent that this node will reach in its merger trees;
5.2. Perl Module for Data Extraction

A Perl module is provided that allows for easy extraction of datasets from the GALACTICUS output file together with a straightforward way to implement derived properties. To use this Perl module, add

```perl
genuse lib "/.perl";
use PDL;
use Galacticus::HDF5;
```
at the start of your Perl script. The Galacticus::HDF5 module will import data from a GALACTICUS HDF5 file into PDL variables. All data are stored in a single structure, which also specifies the file, output and range of trees to read. An example of reading a dataset from a file is:

```perl
givenmy $model;
$model->{file} = "galacticus.hdf5";
$model->{output} = 1;
$model->{tree} = "all";
$model->{dataRange} = [1,2];
$model->{store} = 0;
&HDF5::Get_Dataset($model, ['nodeMass']);
$ datasets = $model->{datasets};
print $ datasets->{nodeMass} . "\n";
given```

The $model object is initialized with information to specify which file, output and trees should be used. Its settable components are:

- **file** The name of the GALACTICUS output file to be read.
- **output** Specify the output number in the file which should be read.
- **tree** Specify the tree which should be read, or use “all” to specify that all trees be read.
- **dataRange** Gives the first and last entry in the dataset to read—this facilitates reading of partial datasets (and therefore reading datasets in a piecemeal fashion). If this component is missing, the entire dataset is read.
- **store** If set to 1, any derived properties will be stored back in the GALACTICUS output file for later retrieval. If set to 0 (or if this option is not present), derived properties will not be stored. Currently, storing of derived properties in the GALACTICUS file is only possible if the **tree** option is set to “all” and no **dataRange** is specified.

The &HDF5::Get_Dataset($model, ['nodeMass']); call requests that the nodeMass dataset be read. It is return as a PDL variable in the nodeMass element of the datasets element which is itself a member of $model. The final lines in the example simply write out the resulting array of nodeMass values.

5.2.1. Derived Properties

Derived properties can be created by giving defining functions along with a regular expression string that allows them to be matched. For example, the Galacticus::Baryons module implements a hot gas fraction property called hotHaloFraction or hotHaloFrac. It has the following form:

```perl
givenpackage Baryons;
use PDL;
use Galacticus::HDF5;
geiven```
5. Extracting and Analyzing Results

use Data::Dumper;

%HDF5::galacticusFunctions = ( %HDF5::galacticusFunctions,
   "hotHalo(Fraction|Frac)" => \&Baryons::Get_hotHaloFraction
 );

my $status = 1;
$status;

sub Get_hotHaloFraction {
   $model = shift;
   $dataSetName = $_[0];
   &HDF5::Get_Dataset($model, ['hotHaloMass', 'nodeMass']);
   $dataSets = $model->{'dataSets'};
   $dataSets->{$dataSetName} = $dataSets->{'hotHaloMass'}/$dataSets->{'nodeMass'};
}

The module begins by adding an entry to the %HDF5::galacticusFunctions hash. The key gives a
regular expression which matches to the name of the property to be defined. The value of the key gives
a reference to a subroutine to be called to evaluate this expression. The subroutine is defined below.
When called, it receives the $model structure along with the name of the requested property. The
subroutine should then simply evaluate the requested property and store it in the appropriate location
within $model. Note that the subroutine can request additional datasets be loaded (as happens above
where hotHaloMass and nodeMass are requested) if they are needed for its calculations.

Available Derived Properties

mergerTreeIndex The index of the merger tree in which the galaxy is found. Provided by: Galacticus::HDF5.

redshift The redshift at which the galaxy exists. Provided by: Galacticus::Time.

time The cosmic time (in Gyr) at which the galaxy exists. Provided by: Galacticus::Time.

expansionFactor The expansion factor at which the galaxy exists. Provided by: Galacticus::Time.

stellarMass The sum of disk and spheroid stellar masses. Provided by: Galacticus::StellarMass.

massColdGas The sum of disk and spheroid cold gas masses. Provided by: Galacticus::GasMass.

starFormationRate The sum of disk and spheroid star formation rates. Provided by: Galacticus::StellarMass.

hostNodeMass For isolated nodes, the node mass. For non-isolated nodes, the mass of the isolated node
in which the node resides. Provided by: Galacticus::HostNode.

stellarMass The sum of disk and spheroid stellar masses (or, whichever of these exist in the model).
   Provided by: Galacticus::StellarMass.

hotHalo(Fraction|Frac) The fraction the node’s mass in the hot gas halo. Provided by: Galacticus::Baryons.

inclination A randomly selected inclination for the disk (in degrees). Provided by: Galacticus::Inclination.
5.2. Perl Module for Data Extraction

^disk|bulge)StellarLuminosity:.*:dustAtlas\([-faceOn\]\)\$ Dust-extinguished luminosities for disk and bulge found by interpolating in the dust tables of Ferrara et al. [1999]. If the [faceOn] qualifier is present, extinctions are computed assuming that the disk is observed face-on, otherwise a random inclination is used. Optionally, the dust atlas file to used can be specified via $dataSet->{'dustAtlasFile'}. The available dust atlases span a limited range of spheroid sizes and central optical depths in their tabulations. Standard behavior is to extrapolate beyond the ends of these ranges. This can be controlled via $dataSet->{'dustAtlasExtrapolateInSize'} and $dataSet->{'dustAtlasExtrapolateInTau'} respectively, which can be set to yes/no (or, equivalently, 1/0). Provided by: Galacticus::DustAttenuation.

^disk|bulge)StellarLuminosity:.*:dustCharlotFall2000$ Dust-extinguished luminosities for disk and bulge found using the model of Charlot and Fall [2000]. Provided by: Galacticus::DustCharlotFall2000.

^totalStellarLuminosity:.*:dustAtlas\([faceOn\]\)$ (Optionally dust-extinguished) luminosities for disk plus bulge found by adding together the corresponding disk and bulge luminosities. Provided by: Galacticus::Luminosities.

^bulgeToTotalLuminosity:.*:dustAtlas\([faceOn\]\)$ Ratio of bulge to total (optionally dust-extinguished) luminosities. Provided by: Galacticus::Luminosities.

^magnitude\([^-]:([^-]):(\d.\d)\):z\(\d.\d\):\(dust[^-]:\)?:vega|AB\)? Absolute magnitude corresponding to a stellar luminosity, in either Vega or AB systems. Provided by: Galacticus::Magnitudes.

^magnitude\([^-]:\)?:vega|AB\)? Absolute magnitude corresponding to the generic luminosity property ^luminosity:$1, in either Vega or AB systems. Provided by: Galacticus::Magnitudes.

^apparentMagnitude\([^-]:\) Apparent magnitude corresponding to the absolute magnitude ^magnitude:$1. Provided by: Galacticus::Magnitudes.

comovingDistance The comoving distance (in Mpc) to the galaxy—provided by Galacticus::Survey (see §5.3.1 for a full description).

luminosityDistance The luminosity distance (in Mpc) to the galaxy—provided by Galacticus::Survey (see §5.3.1 for a full description).

distanceModulus The distance modulus (including the +2.5 log(1 + z) term to account for squeezing of photon frequencies) to the galaxy—provided by Galacticus::Survey (see §5.3.1 for a full description).

redshift The redshift at which the galaxy is observed—provided by Galacticus::Survey (see §5.3.1 for a full description).

angularWeight The weight (in units of ) which should be assigned to this galaxy in order to build a redshift survey—provided by Galacticus::Survey (see §5.3.1 for a full description).

angularDiameterDistance The angular diameterer distance (in Mpc) to the galaxy—provided by Galacticus::Survey (see §5.3.1 for a full description).

^angularPosition\[12]$ The angular position (in radians measured along two orthogonal axes from the center of the field) of the galaxy—provided by Galacticus::Survey (see §5.3.1 for a full description).

^grasilFlux\(\d.\d\)+microns The flux at the given wavelength (specific in microns) of the galaxy as computed by the Grasil code (see §5.5 for a full description).
5. Extracting and Analyzing Results

^grasilInfraredLuminosity^ The total infrared (8–1000µm) luminosity of the galaxy as computed by the Grasil code (see §5.5 for a full description).

^grasilFlux:([^:]*)^ The flux (in Janskys) of the galaxy as computed by the Grasil code integrated under the specified filter (see §5.5 for a full description).

^luminosity:grasil:([^:]*):([^:]*)^ The luminosity (in units of the zero point of the AB magnitude system) of the galaxy as computed by the Grasil code integrated under the specified filter and in the specified frame (see §5.5 for a full description).

flux850micronHayward The flux of the galaxy at 850µm computed using the fitting formula of Hayward et al. [2010], specifically:

\[
\frac{S_{850\mu m}}{Jy} = A \left( \frac{\dot{M}_*}{100M_\odot \text{Gyr}^{-1}} \right)^\alpha \left( \frac{R_{\text{dust}} M_{\text{metals, gas}}}{10^8 M_\odot} \right)^\beta,
\]

where \( R_{\text{dust}} \) is the dust-to-metals ratio, \( \dot{M}_* \) is the total star formation rate in the galaxy and \( M_{\text{metals, gas}} \) is the total mass of metals in the gas phase of the galaxy. Note that the fit given by Hayward et al. [2010] was computed for galaxy at \( z \approx 2 \). The parameters of the fit can be specified by setting elements of $model->\{'haywardSubMmFit'\}: \{'dustToMetalsRatio'\}≡ R_{\text{dust}}, \{'fitNormalization'\}≡ A, \{'starFormationRateExponent'\}≡ \alpha, and \{'dustMassExponent'\}≡ \beta. If these elements are not present the default values of \( A = 0.65 \times 10^{-3}, R_{\text{dust}} = 0.61, \alpha = 0.42 \) and \( \beta = 0.58 \) Hayward et al. [2010] will be used instead. Provided by: Galacticus::SubMmFluxesHayward.

^columnDensity(disk|spheroid)?^ The column density of hydrogen (in units of \( 10^{-23} \) cm\(^{-2} \)) along the line of sight to the center of the galaxy. If a component is specified the calculation is performed for that component, otherwise the sum of disk and spheroid column densities is computed. Provided by: Galacticus::ColumnDensity. For the exponential disk, the column density is given by:

\[
N_H = \frac{X_H}{m_H} \frac{M_{\text{SM}}}{4\pi hr_d^2} \int_0^{\infty} \exp(-r/hr_d) \text{sech}^2\left(z/hr_d\right)dl,
\]

142
where \(r_d\) is the disk scale length, \(h\) is the ratio of vertical scale height to radial scale length, \(X_H\) is the mass fraction in hydrogen, \(m_H\) is the mass of the hydrogen atom and \(l\) is distance along the line of sight. Writing \(z = r / \tan i\) for a disk at inclination \(i\), and \(l = \sqrt{r^2 + z^2} = r \sqrt{1 + 1 / \tan^2 i}\) this becomes:

\[
N_H = \frac{X_H M_{ISM}}{m_H 4\pi h r_d^2} \sqrt{1 + 1 / \tan^2 i} \int_0^\infty \exp(-x) \text{sech}^2(x/h \tan i) dx.
\]

(5.3)

The integral can be evaluated to give:

\[
N_H = \frac{X_H M_{ISM}}{m_H 8\pi h r_d^2} \sqrt{1 + 1 / \tan^2 i} H \left\{ \psi \left( -\frac{H}{4} \right) - \psi \left( \frac{1}{2} - \frac{H}{4} \right) \right\} - 2
\]

(5.4)

where \(H = h \tan i\) and \(\psi(x)\) is the digamma function.

\[\text{peakSFR}\] The peak star formation rate for each galaxy, measured from the \text{starFormationHistories} output group (see §17.4.3). The peak star formation rate reported is therefore that when averaged over the bins used by the star formation history output method (see §17.4.3). Provided by: \text{Galacticus::SFH}.

\[\text{lensingAmplification}\] The gravitational lensing amplification due to large scale structure for each galaxy. The amplification is drawn at random from the redshift-dependent distribution of Takahashi et al. [2011]. Provided by: \text{Galacticus::LensingAmplification}.

\[(\text{disk|spheroid|total})\text{LineLuminosity}:[^:]+(:[^:]+){0,2}:\text{z}[\d.]+\]$ Returns the luminosity of the named emission line, for the named component, at the given redshift in units of Solar luminosities. Optinally, a filter may be provided in which case the emission line luminosity under that filter is returned in units of AB maggies. For example, \text{totalLineLuminosity:balmerAlpha6563:rest:z0.0000} returns the luminosity of the H\(\alpha\) line at \(z = 0\). See §7.5.3 for more details. Provided by: \text{Galacticus::EmissionLines}.

### 5.2.2. Galaxy Clustering via the Halo Model

Galaxy clustering calculations (currently real and redshift space power spectra and two-point correlation functions) can be computed using the \text{Galacticus::HaloModel} Perl module. To use this module, \text{Galacticus} must be run with \text{[outputHaloModelData]=true} (see §14.7) to output data on halo profiles and power spectra. To perform halo model calculations, simply use this module in a Perl script, initialize the data hash, \%dataHash, used for the \text{Galacticus::HDF5} module, and construct a PDL, \$selectedGalaxies, which contains the indices (not the node indices, but the positions within the PDL arrays read in from the \text{Galacticus} output file) of galaxies for which the clustering is to be computed.

A power spectrum can then be computed using:

\[
($\text{waveNumber}$,$\text{linearPowerSpectrum}$,$\text{galaxyPowerSpectrum}$) = &\text{HaloModel::Compute_Power_Spectrum($model,$selectedGalaxies,space => "redshift"});
\]

The PDLs returned contain a list of comoving wavenumbers, the linear power spectrum of matter at the selected time and the (non-linear) power spectrum of the selected galaxies. If the \text{space} option is set to \text{redshift} then a redshift space power spectrum is computed, otherwise a real space power spectrum is computed.

A two-point correlation function can be computed from the returned power spectrum using:

\[
($\text{separations}$,$\text{galaxyCorrelationFunction}$) = &\text{HaloModel::Compute_Correlation_Function($waveNumber,$galaxyPowerSpectrum,$separationMinimum,$separationMaximum,$separationPointsPerDecade});
\]
5. Extracting and Analyzing Results

The first two PDLs are those returned by the power spectrum calculation. The final three give the minimum and maximum separations at which to compute the correlation function and the number of points per decade of separation at which to tabulate the correlation function. The returned PDLs give the comoving separation (in Mpc) and correlation function corresponding to the input power spectrum.

5.3. Topics in Analysis of Galacticus Outputs

5.3.1. Building Volume Limited Samples

The \texttt{mergerTreeWeight} property (see §5.1.6) property specifies the weight to be assigned to each merger tree in a model to construct a representative (i.e. volume limited) sample of galaxies. Galacticus does not typically generate every merger tree in a fixed volume of the Universe (as an N-body simulation might for example) as it’s generally a waste of time to simulate millions of low mass halos and only a small number of high mass halos. The \texttt{mergerTreeWeight} factors correct for this sampling. If merger trees are being built, then the \texttt{mergerTreeWeight}, \( w_i \), for each tree of mass \( M_i \) (where the trees are ranked in order of increasing mass) is given by

\[
    w_i = \int_{M_{\text{min}}}^{M_{\text{max}}} n(M) \, dM, \quad (5.5)
\]

where \( n(M) \) is the dark matter halo mass function and

\[
    M_{\text{min}} = \sqrt{M_{i-1} M_i}, \quad (5.6)
\]

\[
    M_{\text{min}} = \sqrt{M_i M_{i+1}}. \quad (5.7)
\]

Suppose, for example, that we wish to construct a luminosity function of galaxies. In particular, we consider a luminosity bin \( k \) which extends from \( L_k - \Delta k/2 \) to \( L_k + \Delta k/2 \). If tree \( i \) contains \( N_i \) galaxies with luminosities \( l_{i,j} \), where \( j \) runs from 1 to \( N_i \), then the luminosity function in this bin is given by:

\[
    \phi_k = \sum_i \sum_{j=1}^{N_i} \begin{cases} 
        w_i & \text{if } L_k - \Delta k/2 < l_{i,j} \leq L_k + \Delta k/2 \\
        0 & \text{otherwise.} 
    \end{cases} \quad (5.8)
\]

Building Redshift Catalogs

The \texttt{Galacticus::Survey} module provides several derived properties which are useful for constructing redshift surveys, i.e. samples of galaxies distributed in redshift in a way consistent with the chosen cosmology. This module requires a \texttt{Galacticus} model with at least two outputs. The module will first check if \texttt{Galacticus} was run with lightcone output (see §14.8). If it was, the coordinates and redshifts of each galaxy in the lightcone will be used to determine comoving distance, redshift and angular weight. If \texttt{Galacticus} was run without lightcone output then, for each output, it will use the galaxies at that output to populate the range of redshifts lying between the arithmetic mean of the redshift of the output and the redshifts of the preceeding and succeeding outputs (for the latest output the range is extended to \( z = 0 \), while for the earliest output the range is truncated at the redshift of the output itself).

Within this redshift range, galaxies are assigned a comoving distance (property \texttt{comovingDistance}) by selecting at random from the available comoving volume. From this comoving distance a redshift and luminosity distance (properties \texttt{redshift} and \texttt{luminosityDistance} respectively) are determined. Note that galaxies within an individual host halo are not kept spatially co-located—they can each be assigned different comoving distances within the available range. In addition to these properties, the \texttt{Galacticus::Survey} module provides a \texttt{angularWeight} property. This gives the mean number of each galaxy that would be found in a solid angle of one steradian.
5.4. Postprocessing Scripts

5.5. Reprocessing Through Dust Using GRASIL

GALACTICUS computes the star formation histories and, optionally, the luminosities of stellar populations in galaxies. The effects of dust on galaxy spectra is handled by post-processing of GALACTICUS output. A simple treatment of dust-extinction of starlight is described in §5.2.1. For a more detailed treatment of dust extinction, and the re-emission of starlight by dust, GALACTICUS is able to interface with the GRASIL radiative transfer code described by Silva et al. [1998].

To process a GALACTICUS galaxy through GRASIL use the following method:

1. Run GALACTICUS to generate galaxies. GRASIL requires a detailed star formation history for each galaxy it processes. Therefore, you should set [starFormationHistoriesMethod]=metallicity split in your input parameter file. Other parameters controlling the details of the star formation history recording are discussed in §17.4.3. Note that you should ensure that the history is recorded with sufficient precision to permit an accurate calculation by GRASIL. Additionally, you may want to consider using the same stellar population data in GALACTICUS as is used by GRASIL—suitable files in GALACTICUS format can be downloaded from the GALACTICUS web site.

2. Select a galaxy from the output to process through GRASIL. You will need to know the output number, tree index and node index of the galaxy;

3. Run the Extract_Star_Formation_History_for_Grasil.pl script to extract the star formation history for this galaxy in a format suitable for input into GRASIL:

   scripts/aux/Extract_Star_Formation_History_for_Grasil.pl <inputFile> <outputIndex> <treeIndex> <nodeIndex> <grasilFile> [<plotFile>]

   where inputFile is the name of the GALACTICUS model file, outputIndex, treeIndex and nodeIndex are the quantities described above that identify the galaxy of interest and grasilFile is the name of the file to which the star formation history should be written. GRASIL convention dictates that this file should have the suffix .dat. The optional plotFile is the name of a file to which a plot of the star formation history will be written.

4. Create a suitable input parameter file for GRASIL, with the same name as your star formation file created above, but with the suffix .par. An example of such a file is given in aux/Grasil/grasilExample.par—refer to the GRASIL documentation for details of the parameters and how to control GRASIL.

5. Download the GRASIL executable from here and supporting data files from here and unpack them.

6. Run GRASIL:

   aux/Grasil/grasil <fileNameRoot>

   where fileNameRoot is the name of the parameter file you created without the .par suffix. GRASIL will now process (this will probably take a few minutes) the galaxy and output a set of files describing the spectral energy distribution of the galaxy (possibly as viewed from multiple angles depending on your input parameter file). See the GRASIL documentation for full details on the output data.

---

2You can put these files where ever you want. Usually, we place them into aux/Grasil/.
5. Extracting and Analyzing Results

5.5.1. Using the Galacticus::Grasil Module

A more automated way to compute fluxes using Grasil is to use the Galacticus::Grasil Perl module that is provided with GALACTICUS. This model provides additional derived properties in the usual way (see §5.2.1 for details). Currently, observed fluxes are provided, via a derived property grasilFlux<XXX>microns, which will give the observed flux of the galaxy at wavelength \( \lambda = <XXX> \mu m \).

Additionally, the flux integrated under a filter can be found using the derived property grasilFlux:<filter>, where <filter> is the filter name. Luminosities under a filter can be found using luminosity:grasil:<filter>:<frame>, where frame is either rest or observed. Finally, grasilInfraredLuminosity will give the total infrared (8–1000 \( \mu m \)) luminosity of galaxies. Note that these properties require that the Galacticus::Survey module be used to provide redshifts for galaxies (see §5.3.1).

The module will automatically run GRASIL using the parameters given in data/grasilBaseParameters.txt, subject the modifications specified in the grasilOptions element of $model. Allowed options are:

- \$dataSet-{'grasilOptions'}-{'dustToMetalsRatio'}: Sets the dust to metals ratio used in GRASIL;
- \$dataSet-{'grasilOptions'}-{'includePAHs'}: Set to 0/1 to switch off/on calculations of PAH features in GRASIL;
- \$dataSet-{'grasilOptions'}-{'fluctuatingTemperatures'}: Set to 0/1 to switch off/on calculations of fluctuating dust grain temperatures in GRASIL;
- \$dataSet-{'grasilOptions'}-{'wavelengthCount'}: Specifies the number of wavelengths to use in calculating radiative transfer (i.e. the “nlf” parameter in GRASIL);
- \$dataSet-{'grasilOptions'}-{'radialGridCount'}: Specifies the number of radial grid cells to use in calculating radiative transfer (i.e. the “ndr” parameter in GRASIL);
- \$dataSet-{'grasilOptions'}-{'recomputeSEDS'}: If set to 1, SEDs will be computed for all galaxies even if they have been previously computed. (This can be useful to recompute SEDs with different options passed to GRASIL for example.) Set to 0 to re-use previously computed SEDs;
- \$dataSet-{'grasilOptions'}-{'maxThreads'}: Specifies the number of parallel threads to launch, each of which will run an instance of GRASIL. If not specified this number will default to the number of available cores;
- \$dataSet-{'grasilOptions'}-{'cpuLimit'}: Specifies the maximum time (in seconds) for which a Grasil calculation should be allowed to run before being terminated. Defaults to 3600s.

If necessary, the GRASIL code and data files will be downloaded automatically. Where possible, multiple instances of GRASIL are run in parallel to speed up the calculation.

The computed spectral energy distribution (SED) is stored in the HDF5 output file in dataset grasilSEDS/Output<outputIndex>/mergerTree<treeIndex>/node<nodeIndex>/SED, where outputIndex, treeIndex and nodeIndex are respectively the indices of the output, merger tree and node to which the galaxy belongs. The wavelengths and inclinations at which the SED is tabulated are similarly stored in the same group in datasets wavelength and inclination. If the SED has been previously computed for a given galaxy, it will be read from file instead of recomputing using Grasil. The flux is found by interpolating to the relevant rest-frame wavelength and observed inclination.

The Galacticus::Grasil module supports the selection element of $model. If this element is set to contain a PDL giving the selection of galaxies to process then only those galaxies will have their Grasil fluxes computed, rather than all galaxies in the output. Note that if the resulting dataset is stored back to the HDF5 file then any non-selected galaxies will be assigned zero flux, and these zero fluxes will be reported on future attempts to access the flux.\(^3\)

\(^3\)The selection element of $model will be more generally supported in future versions of GALACTICUS and will more elegantly handle storing of partial datasets to file to avoid this problem.
5.6. Meta-Analysis of GALACTICUS

GALACTICUS contains modules which allow it to analyze and profile its own performance.

5.6.1. Tree Construction/Evolution Timing

The Galacticus_Meta_Tree_Timing module records the time taken to construct and evolve each merger tree. Setting [metaCollectTimingData]=true will cause tree timing data to be recorded and output to the metaData/treeTiming group. Three datasets are written to this group:

- treeMasses: Gives the base node masses of the recorded trees (in units of $M_{\odot}$);
- treeConstructTimes: Gives the time (in seconds) taken to construct each merger tree;
- treeEvolveTimes: Gives the time (in seconds) taken to evolve each merger tree.

5.6.2. ODE Evolver Profiler

The Galacticus_Meta_Evolver_Profiler module records statistics on the performance of the main ODE solver used to advance galaxies through time.

Note: Currently, this profiler requires access to features of the GNU Scientific Library that are not implemented within FGSL. As such, this functionality is normally not compiled with GALACTICUS. If you want to use the profiler, contact Andrew Benson and request a copy of the modified FGSL source code. Once this is installed, the profiler can be activated by including -DPROFILE in the compilation options (e.g. add this to your GALACTICUS_FCFLAGS environment variable).

When active, setting [profileOdeEvolver]=true will activate profiling. Each step taken by the ODE evolver is then analyzed. First, a record of the size of the time step taken is recorded. Second, the property which is currently limiting the time step size (i.e. that which has the largest error over the step as judged using the same heuristics as the ODE solver uses to determine step size) is determined and a record of this is kept.

At the end of a run the accumulated data is written to the GALACTICUS output file, into a group named metaData/evolverProfiler. A histogram of time step sizes is written to metaProfileTimeStepCount with bins specified in metaProfileTimeStep—these bins can be adjusted using [metaProfileTimeStepMinimum], [metaProfileTimeStepMaximum] and [metaProfileTimeStepPointsPerDecade]. A histogram of which properties limited step size is written to metaProfilePropertyHitCount with the associated property names written to [metaProfilePropertyNames]. Property names can only be determined if the component to which they belong supports the decodePropertyIdentifiersTask directive (see §17.4.3). Properties which could not be decoded in this way are listed as unknown.

5.7. Meta-Data in Plots

GALACTICUS writes extensive metadata to the XMP section of plots resulting from analysis of GALACTICUS outputs. Metadata written includes all GALACTICUS parameter values, GALACTICUS version and build information, the source code changeset and the model UUID. The intention is to include sufficient metadata that the original model and analysis can be repeated in complete detail. The scripts/aux/extractMetaData.pl script can be used to extract this metadata from a plot file. For example:

scripts/aux/extractMetaData.pl myPlot.pdf myMetaData

will extract the metadata from file myPlot.pdf, writing a report to screen on GALACTICUS version and build information. It will also output the following files:
myMetaDataParameters.xml A GALACTICUS input parameter file containing all parameters used to run the GALACTICUS model from which myPlot.pdf was made;

myMetaDataScript.pl The script used to create myPlot.pdf;

myMetaData.bundle A bundled changeset for Mercurial containing the committed source changeset used to build GALACTICUS. This can be applied to a GALACTICUS checkout using the hg unbundle command;

myMetaData.patch A bundled diff of the GALACTICUS source against the committed source. This can be applied to a GALACTICUS checkout (after applying myMetaData.bundle) using the hg patch command.

5.8. Perl Statistics Modules

GALACTICUS provides some Perl modules which compute useful statistics. These are described below.

5.8.1. Statistics::Histograms

The Statistics::Histograms module computes histograms from a weighted set of points. The module provides a single function, which is used as follows:

```perl
( my $histogram, my $error ) = &Histogram(
   $binCenters,
   $values,
   $weights,
   normalized => 1,
   differential => 1,
   gaussianSmooth => $sigma
);
```

Given a PDL, $binCenters, containing the positions of the bin centers, this function will construct a histogram of the points in the $values PDL, using weights as given by the $weights PDL. The histogram is returned as $histogram with Poisson errors in $errors. The function currently assumes that the bins are uniformly spaced.

The following options are available:

normalized [Default: 0] If set to 1, then the histogram will be normalized to sum to unity;

differential [Default: 0] If set to 1, then the histogram will be divided through by the bin width, to make it differential;

gaussianSmooth [Default: no smoothing] If present, this option must specify a PDL which gives, for each point, the value of \( \sigma \) in a Gaussian smoothing to be applied to that point before it is added to the histogram. As such, each point will contribute a fraction of its weight to each bin in the histogram.
6. Input Data

In some configurations, GALACTICUS requires additional input data to run. For example, if asked to process galaxy formation through a set of externally derived merger trees, then a file describing those trees must be given. In the remainder of this section we describe the structure of external datasets which can be inputs to GALACTICUS.

6.1. Broadband Filters

To compute luminosities through a given filter, GALACTICUS requires the response function, \( R(\lambda) \), of that filter to be defined. GALACTICUS follows the convention of Hogg et al. [2002] in defining the filter response to be the fraction of incident photons received by the detector at a given wavelength, multiplied by the relative photon response (which will be 1 for a photon-counting detector such as a CCD, or proportional to the photon energy for a bolometer/calorimeter type detector. Filter response files are stored in data/filters/. Their structure is shown below, with the SDSS_g.xml filter response file used as an example:

```xml
<filter>
  <description>SDSS g vacuum (filter+CCD +0 air mass)</description>
  <name>SDSS g</name>
  <origin>Michael Blanton</origin>
  <response>
    <datum>3630.000 0.0000000E+00</datum>
    <datum>3680.000 2.2690000E-03</datum>
    <datum>3730.000 5.4120002E-03</datum>
    <datum>3780.000 9.8719997E-03</datum>
    <datum>3830.000 2.9449999E-02</datum>
  </response>
  <effectiveWavelength>4727.02994472695</effectiveWavelength>
  <vegaOffset>0.107430167298754</vegaOffset>
</filter>
```

The `description` tag should provide a description of the filter, while the `name` tag provides a shorter name. The `origin` tag should describe from where/whom this filter originated. The `response` element contains a list of `datum` tags each giving a wavelength (in Angstroms) and response pair. The normalization of the response is arbitrary. The `effectiveWavelength` tag gives the mean, response-weighted wavelength of the filter and is used, for example, in dust attenuation calculations. The `vegaOffset` tag gives the value (in magnitudes) which must be added to an AB-system magnitude in this system to place it into the Vega system. Both `effectiveWavelength` and `vegaOffset` can be computed by running

```
scripts/filters/vega_offset_effective_lambda.pl data/filters
```

which will compute these values for any filter files that do not already contain them and append them to the files.
6.2. Merger Trees

While Galacticus can build merger trees using analytic methods it is often useful to be able to utilize merger trees from other sources (e.g., extracted from an N-body simulation). To facilitate this, Galacticus allows merger trees to be read from an HDF5 file. To do so, set the [mergerTreeConstructMethod] input parameter to read and specify the filename to read via their [mergerTreeReadFileName] parameter.

The HDF5 file should follow the general purpose format described in §A. An example of how to construct such a file can be found in the tests/nBodyMergerTrees folder. In that folder, the getMillenniumTrees.pl script will retrieve a sample of merger trees from the Millennium Simulation database and use the Merger_Tree_File_Maker.exe code supplied with Galacticus to convert these into an HDF5 file suitable for reading into Galacticus. The getMillenniumTrees.pl script requires you to have a username and password to access the Millennium Simulation database. These can be entered manually or stored in a section of the galacticusConfig.xml file (see §3.1) as follows:

```xml
<millenniumDB>
  <host>
    <name>myHost</name>
    <user>myUserName</user>
    <passwordFrom>kdewallet</passwordFrom>
  </host>
  <host>
    <name>default</name>
    <user>myUserName</user>
    <passwordFrom>input</passwordFrom>
  </host>
</millenniumDB>
```

Here, each host section describes rules for a given computer (with “default” being used if no specific match is found). The user element gives the user name to use, while the passwordFrom element specifies how the password should be obtained. Currently allowed mechanisms are “input”, in which case the password is read from standard input, and “kdewallet”, in which case the password is stored in and retrieved from the KDE wallet utility.

6.2.1. Processing of Merger Tree Files

The “read” merger tree construction method (see §17.4.1) reads these files and processes them into a form suitable for Galacticus to evolve. Merger trees are inherently complex structures, particularly when the possibility of subhalos are considered. Galacticus is currently designed to work with single descendent merger trees, i.e., ones in which the tree structure is entirely defined by specifying which node a given node is physically associated with at a later time. Additionally, Galacticus expects the merger tree file to contain information on the host node, i.e., the node within which a given node is physically located. In the following, these two properties are labelled descendentNode and hostNode. Galacticus assumes that nodes for which descendentNode=hostNode are isolated halos (i.e., they are their own hosts) while other nodes are subhalos (i.e., they are hosted by some other node). An example of a simple tree structure is shown in Fig. 6.1. The particular structure would be represented by the following list of nodes and node properties (a −1 indicates that no descendent node exists):

---

1 If you do not have a username and password for the Millennium Simulation database you can request one from contact@g-vo.org.
The following should be noted when constructing merger tree files:

- Note that GALACTICUS does not require that nodes be placed on a uniform grid of times/redshifts, nor that mass be conserved along a branch of the tree. After processing the tree in this way, GALACTICUS builds additional links which identify the child node of each halo and any sibling nodes. These are not required to specify the tree structure but are computationally convenient.

- It is acceptable for a node to begin its existence as a subhalo (i.e. to have never had an isolated node progenitor). Such nodes will be created as satellites in the merger tree and, providing the selected node components (see §12) initialize their properties appropriately, will be evolved correctly.

- It is acceptable for an isolated node to have progenitors, none of which are a primary progenitor. This can happen if all progenitors descend into subhalos in the isolated node. In such cases, GALACTICUS will create a clone of the isolated node at a very slightly earlier time to act as the primary progenitor. This is necessary to allow the tree to be processed correctly, but does not affect the evolution of the tree.

- Normally, cases where a node’s host node cannot be found in the forest will cause GALACTICUS to exit with an error. Setting [mergerTreeReadMissingHostsAreFatal]=false will instead circumvent this issue by making any such nodes self-hosting (i.e. they become isolated nodes rather than subhalos). Note that this behavior is not a physically correct way to treat such cases—it is intended only to allow trees to be processed in cases where the full forest is not available.

- It is acceptable for nodes to jump between branches in a tree, or even to jump between branches in different trees. In the latter case, all trees linked by jumping nodes (a so-called “forest” of connected trees) must be stored as a single tree (with multiple root-nodes) in the merger tree file. GALACTICUS will process this forest of trees simultaneously, allowing to nodes to move between their branches.

- It is acceptable for a subhalo to later become an isolated halo (as can happen due to three-body interactions; see Sales et al. 2007). If [mergerTreeReadAllowSubhaloPromotions]=true then such cases will be handled correctly (i.e. the subhalo will be promoted back to being an isolated halo). If [mergerTreeReadAllowSubhaloPromotions]=false then subhalos are not permitted to become isolated halos. In this case, the following logic will be applied to remove all such cases from the tree:

   → For any branch in a tree which at some point is a subhalo:

   → Beginning from the earliest node in the branch that is a subhalo, repeatedly step to the next descendent node;

   → If that descendent is not a subhalo then:
Figure 6.1.: An example of a simple merger tree structure. Colored circles represent nodes in the merger tree. Each node has a unique index indicated by the number inside each circle. Black arrows link each node to its descendent node (as specified by the `descendentNode` property. Where a node is not its own host node it is placed inside its host node.
→ If there is not currently any non-subhalo node which has the present node as its descendent then current node is only descendent of a subhalo. Therefore, try to make this node a subhalo, and propose the descendent of the host node of the previous node visited in the branch as the new host:

→ If the proposed host exists:

→ If the mass of the current node is less than that of the proposed host:

→ If the proposed hosts exists before the current node, repeatedly step to its descendants until one is found which exists at or after the time of the current node. This is the new proposed host.

→ If the proposed host is a subhalo, make it an isolated node.

→ The current node is made a subhalo within the proposed host.

→ Otherwise:

→ The current node remains an isolated node, while the proposed host is instead made a subhalo within the current node.

→ Otherwise:

→ The proposed host does not exists, which implies the end of a branch has been reached. Therefore, flag the current node as being a subhalo with a host identical to that of the node from which it descended.

Requirements for Galacticus Input Parameters

The following requirements must be met for the input parameters to Galacticus when using merger trees read from file:

• The cosmological parameters ($\Omega_M$, $\Omega_A$, $\Omega_b$, $H_0$, $\sigma_8$), if defined in the file, must be set identically in the Galacticus input file unless you set [mergerTreeReadMismatchIsFatal]=false in which case you’ll just be warned about any mismatch;

• Galacticus assumes by default that all merger trees exist at the final output time—if this is not the case set [allTreesExistAtFinalTime]=false.

6.2.2. Setting of Halo Properties

Dark Matter Scale Radii

If [mergerTreeReadPresetScaleRadii]=true and the halfMassRadius dataset is available within the haloTrees group (see §A.6) then the half-mass radii of nodes will be used to compute the corresponding scale length of the dark matter halo profile\(^2\). This requires a dark matter profile scale component which supports setting of the scale length (see §12.13).

\(^2\)The scale radius is found by seeking a value which gives the correct half mass radius. It is therefore important that the definition of halo mass (specifically the virial overdensity) in Galacticus be the same as was used in computing the input half mass radii.
6. Input Data

**Satellite Merger Times**

If \[\text{mergerTreeReadPresetMergerTimes} = \text{true}\] then merger times for satellites will be computed directly from the merger tree data read from file. When a subhalo has an isolated halo as a descendent it is assumed to undergo a merger with that isolated halo at that time. Note that this requires a satellite orbit component method which supports setting of merger times (e.g. \[\text{treeNodeMethodSatelliteOrbit} = \text{preset}\]).

**Dark Matter Halo Spins**

If \[\text{mergerTreeReadPresetSpins} = \text{true}\] and the \text{angularMomentum} dataset is available within the \text{haloTrees} group (see §A.6) then the spin parameters of nodes will be computed and set. This requires a dark matter halo spin component which supports setting of the spin (see §12.12).
7. Tutorials

This chapter contains step-by-step guides to performing common tasks with GALACTICUS.

7.1. Running GALACTICUS on N-body Merger Trees

See §A.10 for details of how to build merger tree files suitable for input into GALACTICUS.

7.1.1. Setting Input Parameters

To utilize merger trees from the file that you created in a GALACTICUS run it’s necessary to set two parameters in the input parameter file that you will use for the run:

<!-- Specify that merger trees are to be read from file, and give the name of the file to read -->
<parameter>
   <name>mergerTreeConstructMethod</name>
   <value>read</value>
</parameter>
<parameter>
   <name>mergerTreeReadFileName</name>
   <value>myNBodyTrees.hdf5</value>
</parameter>

The first of these [mergerTreeConstructMethod]=read tells GALACTICUS that merger trees will be constructed by reading them from a file. The second, [mergerTreeReadFileName], gives the name of the file from which to read the trees. In this example, we use the name of the file that was just created.

In addition to specifying that trees should be read from a file, it’s also important to ensure that the values of cosmological parameters in GALACTICUS match those in the merger tree file. (If they don’t match, GALACTICUS will stop with an error message unless you set [mergerTreeReadMismatchIsFatal]=false in which case you’ll just be warned about any mismatch.) In our case of using merger trees from the Millennium Simulation, the correct cosmological parameter values can be set as follows:

<!-- Use Millennium Simulation cosmology. -->
<parameter>
   <name>H_0</name>
   <value>73.0</value>
</parameter>
<parameter>
   <name>Omega_Matter</name>
   <value>0.25</value>
</parameter>
<parameter>
   <name>Omega_DE</name>
   <value>0.75</value>
</parameter>
Normally, Galacticus assumes that all merger trees will exist (i.e. have at least one node present) at the final output time. This may not be true of trees extracted from an N-body simulation—in this case Galacticus can be informed of this fact by setting:

```xml
<parameter>
  <name>allTreesExistAtFinalTime</name>
  <value>false</value>
</parameter>
```

N-body merger trees are often built from “snapshots” of the simulation, i.e. all of the nodes exist at a set of discrete times. Often we want to output nodes at precisely these output times. In such cases it is useful to set:

```xml
<parameter>
  <name>mergerTreeReadOutputTimeSnapTolerance</name>
  <value>1.0d-3</value>
</parameter>
```

which ensures that the times of nodes are adjusted to lie at precisely the output time if that time is within the specified relative tolerance (this avoids any small differences between node times and output times that can arises due to rounding errors when converting from redshifts to times and vice-versa).

Further parameters can be set to control what information from the stored trees will be used in Galacticus. Examples are given below.

**Node Positions**

If position and velocity information for tree nodes is available within the merger tree file then Galacticus can be instructed to use this information by using the “preset” method for tree node positions and telling the merger tree construction method to preset node positions as follows:

```xml
<!-- Use merger tree node positions -->
```
Running Galacticus on N-body Merger Trees

<parameter>
    <name>treeNodeMethodPosition</name>
    <value>preset</value>
</parameter>

<parameter>
    <name>mergerTreeReadPresetPositions</name>
    <value>true</value>
</parameter>

If position information is unavailable, the “null” position method can be selected and the merger tree construction method instructed not to preset positions as follows:

<!-- Do not use merger tree node positions -->
<parameter>
    <name>treeNodeMethodPosition</name>
    <value>null</value>
</parameter>

<parameter>
    <name>mergerTreeReadPresetPositions</name>
    <value>false</value>
</parameter>

Virial Orbits

If position and velocity information for tree nodes is available within the merger tree file then Galacticus can be instructed to use this information to estimate the orbit of each subhalo at the point at which it crosses the virial radius of its host halo. This “virial orbit” may then be used by, for example, calculations of merging timescales.

<!-- Use merger tree node positions to compute orbits at the virial radius -->
<parameter>
    <name>mergerTreeReadPresetOrbits</name>
    <value>true</value>
</parameter>

<parameter>
    <name>mergerTreeReadPresetOrbitsBoundOnly</name>
    <value>true</value>
</parameter>

<parameter>
    <name>mergerTreeReadPresetOrbitsSetAll</name>
    <value>true</value>
</parameter>

<parameter>
    <name>mergerTreeReadPresetOrbitsAssertAllSet</name>
    <value>true</value>
</parameter>

Typically, a merging halo is not seen at precisely the time at which it crosses the virial radius of its host (due to the fact that N-body simulations are output at discretely spaced timesteps). Therefore, Galacticus computes the orbit at the time just prior to merging and assumes that the orbital parameters (energy and angular momentum) remain fixed to propagate the orbit to the virial radius of the host. The second parameter in the above example, [mergerTreeReadPresetOrbitsBoundOnly], specifies whether or not...
only bound orbits should be set. Some calculations (e.g. of subhalo merging times) assume bound orbits and may fail if given an unbound orbit. Setting this option to true causes only bound orbits to be preset—unbound orbits are ignored. Note that some orbits cannot be propagated to the virial radius (i.e. their pericenter is larger than the virial radius). The [mergerTreeReadPresetOrbitsSetAll] option, if true, will cause such orbits to be assigned randomly using the selected XXXXX method, such that all orbits are assigned. The [mergerTreeReadPresetOrbitsAssertAllSet] option requires that all orbits be set—if [mergerTreeReadPresetOrbitsSetAll]=false and [mergerTreeReadPresetOrbitsAssertAllSet]=true then GALACTICUS will exit with an error message if any orbit cannot be set.

Merging Times and Targets

The times at which subhalos merge with their host halo can be determined directly from the merger tree file if subhalo information is included in that file. Merging is assumed to occur when the subhalo no longer has a distinct descendent (i.e. it descends into a non-subhalo). If merging times are to be computed in this way set

```
<parameter>
  <name>treeNodeMethodSatellite</name>
  <value>preset</value>
</parameter>
<parameter>
  <name>mergerTreeReadPresetMergerTimes</name>
  <value>true</value>
</parameter>
```

which select a satellite orbit method that allows merger times to be present and tell the merger tree construction method to preset those merger times respectively. If merger times are not to be computed in this way then instead set, for example,

```
<parameter>
  <name>treeNodeMethodSatellite</name>
  <value>standard</value>
</parameter>
<parameter>
  <name>mergerTreeReadPresetMergerNodes</name>
  <value>false</value>
</parameter>
<parameter>
  <name>satelliteMergingMethod</name>
  <value>Jiang2008</value>
</parameter>
```

which selects a standard satellite orbit method, prevents attempts to preset the merger times and selects the Jiang2008 method for computing merger times instead.

In addition to setting the times of merger events, it is possible to set the target node with which a merging node should merge. By default, GALACTICUS will assume that all merging occurs with the non-subhalo host node in which a subhalo is located. This may not be the desired behavior when using N-body merger trees. For example, such trees may indicate that a subhalo merges with another subhalo. Setting
7.1. Running GALACTICUS on N-body Merger Trees

will cause the target node with which each merger should occur to be determined from the merger tree structure and preset for use in GALACTICUS.

It is possible to add a delay between the last time at which a subhalo was seen in a simulation and the time at which it is considered to merge. This functionality is motivated by the consideration that a subhalo vanishing from a simulation may be simply due to it dropping below resolution rather than it actually having undergone a merger. The parameter `mergerTreeReadSubresolutionMergingMethod` can be used to select a satellite merging timescale method (see §13.44.1) to use in this case. (It is set by default to “null” such that no delay before merging occurs.) The orbit of the subhalo around its parent at the last time it is present in the merger tree is passed to this method and used to estimate a time until merging. This delay is added to the time at which the subhalo merges and, if merge target nodes are being set, the target node is updated accordingly.

**Subhalo Indices**

The indices of subhalos are usually frozen at the index of the halo just prior to becoming a subhalo. The index of the corresponding halo in the original tree (as read from file) can be tracked as follows:

```xml
<parameter>
    <name>treeNodeMethodSatellite</name>
    <value>preset</value>
</parameter>
<parameter>
    <name>mergerTreeReadPresetSubhaloIndices</name>
    <value>true</value>
</parameter>
```

to first select the “preset” satellite orbit method (which allows subhalo indices to be preset) and, second, to instruct the merger tree construction algorithm to preset those indices. The index will then be available in output as `satelliteNodeIndex`.

**Subhalo Masses**

The masses of subhalos (specifically their time evolution after they become subhalos) can be set using the values stored in the merger tree file (if available). To set subhalo masses in this way use

```xml
<parameter>
    <name>treeNodeMethodSatellite</name>
    <value>preset</value>
</parameter>
<parameter>
    <name>mergerTreeReadPresetSubhaloMasses</name>
    <value>true</value>
</parameter>
```

to first select the “preset” satellite orbit method (which allows subhalo masses to be preset) and, second, to instruct the merger tree construction algorithm to preset those masses.
7. Tutorials

Node Spins

If information on the angular momenta of nodes is available in the merger tree file, this can be used to preset the value of the spin parameter in each node\(^1\) by setting:

```
<parameter>
  <name>mergerTreeReadPresetSpins</name>
  <value>true</value>
</parameter>
```

The spin parameter is set using the spin of each node if available, or otherwise using the angular momentum of each node stored in the merger tree file using:

\[
\lambda = \frac{|\mathbf{J}| ||E||^{1/2}}{GM^{5/2}}
\]  
(7.1)

where \(|\mathbf{J}|\) is the magnitude of the node’s angular momentum, \(M\) is the node’s mass and \(E\) is its energy. Additionally, by setting:

```
<parameter>
  <name>mergerTreeReadPresetSpins3D</name>
  <value>true</value>
</parameter>
```

the spin vector of each node will be set (assuming that the vector spin or angular momenta of nodes are available in the merger tree file) using:

\[
\lambda = \frac{\mathbf{J} ||E||^{1/2}}{GM^{5/2}}.
\]  
(7.2)

If spins could not be determined for some halos the spin (or angular momentum) should be set to zero in the merger tree file, and the parameter [mergerTreeReadPresetUnphysicalSpins] set to true. GALACTICUS will then assign a spin to such halos by sampling from the selected spin distribution (see §13.8.7).

Node Scale Radii

If information on the half-mass or scale radii of nodes is available in the merger tree file, it can be used to preset the value of the dark matter halo scale radius in each node by setting:

```
<parameter>
  <name>mergerTreeReadPresetScaleRadii</name>
  <value>true</value>
</parameter>
```

Before doing this, it is important to be sure that the half-mass or scale radii of the nodes are reliable. For example, in low mass nodes extracted from an N-body simulation resolution effect may limit the accuracy of the measured half-mass or scale radius. In such cases, use the [mergerTreeReadPresetScaleRadiiMinimumMass] parameter to specify the lowest mass halos for which the scale radii should be preset—lower mass halos will be assigned a scale radius using the method specified by the [mergerTreeReadConcentrationFallbackMethod] parameter (which will default to the value of [darkMatterConcentrationMethod]; see §13.8.3). It is also possible to specify minimum and maximum allowed concentrations when computing the scale radius.

---

\(^1\)Before doing this, it is important to be sure that the angular momenta of the nodes are reliable. For example, in low mass nodes extracted from an N-body simulation resolution effect may limit the accuracy of the measured angular momentum.
from the half mass radius using the \texttt{mergerTreeReadPresetScaleRadiiConcentrationMinimum} and \texttt{mergerTreeReadPresetScaleRadiiConcentrationMaximum} parameters. If matching the half mass radius would require a concentration outside of this range, \textsc{Galacticus} will abort unless \texttt{mergerTreeReadPresetScaleRadiiFailureIsFatal} = \texttt{false}, in which case it will instead silently use the fallback concentration method described above.

If only half-mass radii are available, the scale radius is set by using a root finding algorithm to ensure that half of the total halo mass is enclosed within the specified half-mass radius.

### Miscellaneous N-body Properties

Several miscellaneous properties often available from N-body merger trees can also be preset by setting the following parameters to \texttt{true}:

- \texttt{mergerTreeReadPresetParticleCounts} Sets the number of particles in each halo (requires the \texttt{particleCount} dataset to be present in the merger tree file);
- \texttt{mergerTreeReadPresetVelocityMaxima} Sets the maxima of halo rotation curves (requires the \texttt{velocityMaximum} dataset to be present in the merger tree file);
- \texttt{mergerTreeReadPresetVelocityDispersions} Sets the velocity dispersion of halos (requires the \texttt{velocityDispersion} dataset to be present in the merger tree file).

### “Fly-by” Halos

In some cases, a halo that is part of one tree can later become part of another tree. This can happen in so-called “fly-by” encounters where a halo may briefly become a subhalo in a halo in tree A then leave that halo and become a subhalo in tree B.

The correct way to handle this issue is to combine trees A and B into a single tree (which will now have multiple base nodes). \textsc{Galacticus} will then process these two trees simultaneously, correctly handling the fly-by, and outputting the trees as two separate trees.

If for some reason this is not possible or desired, the fly-by problem will normally cause \textsc{Galacticus} to complain that the host halo of a node cannot be found (since it exists in a different tree). This problem can be avoided by setting:

```xml
<parameter>
  <name>mergerTreeReadMissingHostsAreFatal</name>
  <value>false</value>
</parameter>
```

In this case, nodes with missing hosts are simply treated as being isolated halos. This will avoid an error condition, but is not a physically correct way to handle such cases, so use with caution.

### 7.1.2. Analyzing the Output

#### Positions and Velocities

Components of the position of each node are output as \texttt{positionX}, \texttt{positionY} and \texttt{positionZ} and can be accessed in the same way as other output properties from \textsc{Galacticus} (see §5.1.6 and §5.2).

#### Subhalo Masses

The current mass of subhalos is available via the \texttt{nodeBoundMass} output dataset and can be accessed in the same way as other output properties from \textsc{Galacticus} (see §5.1.6 and §5.2). For non-subhalos this property is equal to the usual \texttt{nodeMass} property.
7. Tutorials

7.2. Generating Mock Catalogs with Lightcones from the Millennium Simulation

Suppose that you want to create a catalog of galaxies as would be found in a survey of an area of the sky out to some redshift. Such a “mock catalog” can be built by populating with galaxies all of the dark matter halos which happen to lie within the cone which that area makes as it is projected from the observer through the Universe.

Generating such a mock catalog using GALACTICUS involves first extracting the halos (and their merger trees) within this “lightcone” from a suitable N-body simulation, and then processing them through GALACTICUS. In this tutorial, we will specifically make use of the Millennium Simulation database to provide the merger trees, but the same principles apply to any N-body simulation.

The script, scripts/aux/Millennium_Lightcone_Grab.pl can be used to retrieve merger trees that intersect a given lightcone from the Millennium Database and to store them in GALACTICUS’s format (see §A). The script is used as follows:

```
scripts/aux/Millennium_Lightcone_Grab.pl <lightconeDirectory> <fieldSize> <maximumRedshift> --user <myUserName> --password <myPassword> --treesPerFile <treesPerFile>
```

Here, `<lightconeDirectory>` is the name of a (pre-existing) directory into which merger tree data will be stored, `<fieldSize>` is the length (in degrees) of one side of the square field of view of the lightcone, `<maximumRedshift>` is the highest redshift for which halos should be included in the catalog. The `-user` and `-password` options allow you to specify your username and password for accessing the Millennium Simulation database. Finally, the `-treesPerFile` specifies how many merger trees should be stored in each file (the script will split the lightcone between many files—this is primarily so that each request sent to the Millennium Database server is not too large). If no value is specified a default of 200 trees per file will be used.

The script generates multiple SQL queries to the Millennium database in order to first find all halos which intersect the lightcone and second to retrieve the complete merger tree associated with each such halo. These merger trees are then stored in GALACTICUS’s merger tree file format in files named Lightcone_Trees_AAA:BBB.hdf5 in the given `<lightconeDirectory>`, where AAA and BBB are numbers giving the first and last trees in the file.²

Each of the merger tree files created can then be run through GALACTICUS in the usual way (see §7.1). Outputs should be requested at every Millennium snapshot (up to the largest redshift to be considered), and the lightcone filter should be used to cause only those galaxies which intersect the lightcone to be output—for example:

```
<!-- Set output redshifts to the snapshots in the milliMillennium. -->
<parameter>
  <name>outputRedshifts</name>
  <value>
    0.0000 0.0199 0.0414 0.0645 0.0893 0.1159 0.1444 0.1749 0.2075 0.2425
    0.2798 0.3197 0.3623 0.4079 0.4566 0.5086 0.5642 0.6236 0.6871 0.7550
    0.8277 0.9055 0.9887 1.0779 1.1734 1.2758 1.3857 1.5036 1.6303 1.7663
    1.9126 2.0700 2.2395 2.4220 2.6189 2.8312 3.0604 3.3081 3.5759 3.8657
  </value>
</parameter>
```

```
<!-- Add a lightcone filter with the required geometry -->
```

²Note that these are not the ID numbers of the trees, just a sequential count of all trees retrieved.
7.3. Using the Instantaneous Recycling Approximation

Choosing \( \texttt{stellarPopulationPropertiesMethod}=\textit{instantaneous} \) will cause GALACTICUS to use the instantaneous recycling approximation for all calculations of stellar populations. The recyling rate and yield to use are set by the \([\texttt{imfNAMERecycledInstantaneous}]\) and \([\texttt{imfNAMEYieldInstantaneous}]\) parameters respectively, where \texttt{NAME} is the name of the appropriate IMF.

Setting \( \texttt{stellarPopulationPropertiesMethod}=\textit{noninstantaneous} \) causes GALACTICUS to use a fully non-instantaneous, metal-dependent calculation of recycling, metal production and SNe rates. However, it is possible to force this method to operate in the instantaneous recycling approximation limit (which can be useful for testing and comparison) by setting:

\[
\begin{align*}
\text{<parameter>}
&\quad \text{<!-- Force the calculation of recycling, yields etc. to }\text{ be done assuming instantaneous recycling} \quad -->
&\quad \text{<name>starFormationImfInstantaneousApproximation</name>}
&\quad \text{<value>true</value>}
\end{align*}
\]

\[
\begin{align*}
\text{<parameter>}
&\quad \text{<!-- Set the mass of stars which should be used as the }\text{ dividing line between long-lived and instantaneously }\text{ evolving in this approximation.} \quad -->
&\quad \text{<name>starFormationImfInstantaneousApproximationMassLongLived</name>}
&\quad \text{<value>1.0</value>}
\end{align*}
\]

\[
\begin{align*}
\text{<parameter>}
&\quad \text{<!-- Set the effective age of populations to use in this approximation when computing SNe numbers.} \quad -->
&\quad \text{<name>starFormationImfInstantaneousApproximationEffectiveAge</name>}
&\quad \text{<value>13.8</value>}
\end{align*}
\]
7.4. Computing Dust Attenuation and Emission Using Galacticus+Grasil

**Galacticus** can interface with the **Grasil** code to compute the attenuation of starlight by dust, along with the re-emission of absorbed energy by that dust. To do this, it is necessary to store the entire star formation history of galaxies in a **Galacticus** model, as **Grasil** uses this information to determine the attenuation of stellar populations as a function of their age.

Recording star formation histories is as simple as setting the `[starFormationHistoriesMethod]` parameter to `metallicitySplit`. This particular star formation history method stores the star formation in each galaxy as a function of time and metallicity, as required by **Grasil**. The level of detail with which the star formation history is stored is controlled by several parameters:

- **starFormationHistoryTimeStep** The timestep used in discretizing star formation histories.
- **starFormationHistoryFineTimeStep** The timestep to use in discretizing star formation histories just prior to output times. This should typically be smaller than `[starFormationHistoryTimeStep]` to give improved resolution in the star formation history for recently formed stars.
- **starFormationHistoryFineTime** The period before each output for which the `[starFormationHistoryFineTimeStep]` should be used.
- **starFormationHistoryMetallicityCount** The number of bins in metallicity to use when discretizing the star formation history.
- **starFormationHistoryMetallicityMinimum** The upper limit to the metallicity in the lowest metallicity bin (i.e. the lowest metallicity bin will extend from zero to this value).
- **starFormationHistoryMetallicityMaximum** The upper limit to the metallicity in the highest metallicity bin.

Default values set a timestep of 0.1 Gyr, with 0.01 Gyr timesteps for 0.1 Gyr before each output, along with 10 metallicity bins ranging from $10^{-4} Z_\odot$ to $10 Z_\odot$. It is always recommended to check that these values result in a sufficiently well-resolved star formation history for your purposes.

When run with these parameter settings **Galacticus** will output an additional group to the output file called `starFormationHistories`. This contains a hierarchically arranged set of datasets describing the star formation histories. The hierarchy extends through output number, and merger tree index. For example `starFormationHistories/Output5/mergerTree3/` will contain the star formation history for merger tree 3 at output 5. This group will, in general, contain many datasets, e.g.

- `diskSFH7819` Dataset `{11, 34}`
- `diskTime7819` Dataset `{34}`
- `spheroidSFH7819` Dataset `{11, 34}`
- `spheroidTime7819` Dataset `{34}`

In this case, datasets are present for both a disk and bulge component of node 7819. (If a node does not contain one of these components, the corresponding dataset will be missing.) The “Time” datasets give the times at which the star formation history is stored, while the “SFH” datasets give the mass of stars formed in each time/metallicity bin. (The metallicities themselves are available in the `starFormationHistories/metallicities` dataset.)

Properties of the **Grasil** SED can now be accessed using the **Galacticus**:**Grasil** module (see §5.5). When such properties are requested, **Grasil** will be automatically run on each selected galaxy, the SED
7.4. Computing Dust Attenuation and Emission Using Galacticus+Grasil

computed and stored to the GALACTICUS file\(^3\), and the relevant fluxes computed. If necessary, GRASIL and its data files will be downloaded automatically. Multiple GRASIL models will be run simultaneously if multiple cores are available.

For example:

```perl
# Specify model.
my $galacticus;
$galacticus->{\'file\'} = "/galacticus.hdf5";
$galacticus->{\'store\'} = 0;
$galacticus->{\'tree\'} = "all";

# Specify Grasil options.
$galacticus->{\'grasilOptions\'}->{\'includePAHs\'} = 1;
$galacticus->{\'grasilOptions\'}->{\'fluctuatingTemperatures\'} = 1;
$galacticus->{\'grasilOptions\'}->{\'wavelengthCount\'} = 1000;
$galacticus->{\'grasilOptions\'}->{\'radialGridCount\'} = 30;
$galacticus->{\'grasilOptions\'}->{\'recomputeSEDs\'} = 0;

# Read results from model.
&HDF5::Select_Output($galacticus,2.0);
&HDF5::Get_Dataset ($galacticus,

    [ 
        \'grasilFlux850microns\',
        \'grasilFlux250microns\',
        \'grasilFlux350microns\',
        \'grasilFlux500microns\',
        \'grasilInfraredLuminosity\'
    ]
)
```

The \texttt{grasilOptions} block in the above controls the behavior of GRASIL. Here we’ve chosen to include calculations of polycyclic aromatic hydrocarbon (PAH) features, have accounted for fluctuating temperatures in small grains (both of which slow down the calculation but make it more accurate), have specified the number of wavelengths and the size of the radial grid used to model each galaxy. We have also specified that the SED should not be recomputed—if GRASIL fluxes are requested in future for galaxies in this model, they will be computed from the stored GRASIL SED. If you want to change the parameters of the Grasil calculation then set the \texttt{recomputeSED} option to 1 instead\(^4\). Fluxes are returned in units of Janskys, while the total infrared luminosity (\texttt{grasilInfraredLuminosity}) is returned in units of Solar luminosities.

A simple plotting script is provided which illustrates how to access and use the GRASIL SEDs stored in GALACTICUS files. For example:

```bash
scripts/plotting/plotGrasilSpectrum.pl galacticus.hdf5 5 9 217 43.2 SED.pdf
```

will plot the SED of node 217, in merger tree 9, at output 5 from the \texttt{galacticus.hdf5} file. The SED will be shown for an inclination of 43.2° and the plot will be written to \texttt{SED.pdf}.

\(^3\)GRASIL SEDs are stored in the \texttt{grasilSEDs} group in a hierarchy of output, merger tree, and node groups, as for the star formation histories. Within each node group three datasets are stored, giving the wavelength, inclination, and SED of the galaxy.

\(^4\)Exercise caution when using this option. Recomputing SEDs requires deleting the old SED group. The HDF5 library currently does not clean up the space occupied by the datasets in this deleted group, so the file size can grow rapidly if you repeatedly recompute GRASIL SEDs.
7.5. Outputting Stellar Luminosities

GALACTICUS can compute the stellar luminosity of galaxies in any required combination of filter, redshift, and frame. To cause luminosities to be computed add something such as the following to your input parameter file:

```
<parameter>
  <name>{luminosityFilter}</name>
  <value>SDSS_r</value>
</parameter>
<parameter>
  <name>{luminosityRedshift}</name>
  <value>0.1</value>
</parameter>
<parameter>
  <name>{luminosityType}</name>
  <value>observed</value>
</parameter>
```

This would result in a dataset called `diskLuminositiesStellar:SDSS_r:observed:z0.1000` being output (along with a similar dataset for the spheroid component), corresponding to the luminosity in the SDSS r-band filter, as observed at $z = 0.1$. To get the same filter but in the rest-frame of the galaxy, change the `luminosityType` to "rest". If instead of specifying a unique redshift you instead specify "all" for a filter in the `luminosityRedshift` element, then the filter will be replicated to all output redshifts.

Available filters can be found in the `data/filters` folder. Additionally, it is possible to specify top-hat filters (which have unit transmission over a range of wavelengths) without the need to create a filter file for them. A filter named `topHat_L_R` will be interpreted as a top-hat filter where $L = \lambda$ is the central wavelength (in Angstroms) and $R = R$ is the resolution. A top-hat filter is constructed such that the transmission is unity between $\lambda_1$ and $\lambda_2$, and zero outside that range, and such that $\lambda_1 \lambda_2 = \lambda^2$, $\lambda_2 - \lambda_1 = \lambda/R$. Furthermore, a filter specified as `topHat_Llow_Lhigh_R` will be expanded into a set of contiguous top-hat filters with resolution $R$ spanning the wavelength range $Llow$ to $Lhigh$ with the first filter's lower wavelength limit aligned with $Llow$ and continuing until the central wavelength of the filter is no longer below $Lhigh$.

Luminosities are always output in units of the zero-point of the AB magnitude system, such that $-2.5 \log_{10} L$ (where $L$ is the output luminosity) gives the AB absolute magnitude of the galaxy.

You can add additional luminosities by simply adding more entries in these parameter values. For example:

```
<parameter>
  <name>{luminosityFilter}</name>
  <value>SDSS_r SDSS_g</value>
</parameter>
<parameter>
  <name>{luminosityRedshift}</name>
  <value>0.1 0.3</value>
</parameter>
<parameter>
  <name>{luminosityType}</name>
  <value>observed rest</value>
</parameter>
```
7.5. Outputting Stellar Luminosities

would result in datasets diskLuminositiesStellar:SDSS_r:observed:z0.1000 and diskLuminositiesStellar:SDSS_g:rest:z0.3000.

7.5.1. Postprocessing of Stellar Spectra

Stellar luminosities are computed by convolving a library of simple stellar populations with the star formation history of each galaxy. GALACTICUS allows the spectra of those simple stellar populations to be postprocessed (after being read from file or internally generated for example) before they are utilized in the convolution integral. This postprocessing can modify the spectra in arbitrary ways that depend on wavelength, redshift, and age of stellar population. Furthermore, GALACTICUS allows you to chain together stellar spectra postprocessors into a set to allow multiple postprocessings to be applied. Furthermore again, you can define an arbitrary number of sets and apply different sets to different luminosities.

Typical uses of stellar spectra postprocessors include accounting for absorption of galaxy light by the intervening IGM, or capturing only the light from recent star formation. A full list of the available postprocessors can be found in §13.42.

If you don’t specify a postprocessing set, the “default” set (consisting of the inoue2014 postprocessor; see §13.42.1) is applied to each luminosity calculation. To specify other postprocessing sets add the following to your parameter file:

```xml
<parameter>
  <name>luminosityFilter</name>
  <value>default recent unabsorbed recentUnabsorbed</value>
</parameter>
```

where one set must be specified for each luminosity specified in the luminosityFilter parameter. Note that set names can be reused in order to apply the same postprocessor set to multiple luminosities.

The chain of postprocessors to apply for each set is then specified as follows:

```xml
<parameter>
  <name>stellarPopulationSpectraPostprocessRecentMethods</name>
  <value>inoue2014 recent</value>
</parameter>
<parameter>
  <name>stellarPopulationSpectraPostprocessUnabsorbedMethods</name>
  <value>identity</value>
</parameter>
<parameter>
  <name>stellarPopulationSpectraPostprocessRecentUnabsorbedMethods</name>
  <value>recent</value>
</parameter>
```

In this case we’ve constructed three new sets, in addition to the default set (which applies just the inoue2014 postprocessor). The recent set applies both the inoue2014 IGM absorption postprocessor, followed by the recent postprocessor to retain only recently emitted light. The unabsorbed set ignores IGM absorption entirely—it does this by using the identity postprocessor which leaves the spectrum unaffected. Finally, the recentUnabsorbed set applies only the recent filter while ignoring IGM absorption.

In this way it is relatively easy to extract multiple different measures of luminosity from a GALACTICUS model. For example, you could construct four postprocessor sets, each corresponding to one of the four

---

5Perhaps so that additional dust extinction can be applied to the light of recently formed stars.
different IGM absorption models (lycSuppress, madau1995, meiksin2006, and inoue2014) and apply these to the same luminosity filter to assess how luminosity depends on the IGM model used.

### 7.5.2. Migrating Parameter Files to a New Version

The names and allowed values of parameters often change between versions of GALACTICUS. To permit easy and error-free migration between versions a script is provided to translate parameter files from earlier to later versions. To migrate a parameter file simply use:

```
scripts/aux/parametersMigrate.pl parameters.xml newParameters.xml
```

By default, this script will translate from the previous to the current version of GALACTICUS. If your parameter file contains a version element then this will be used to determine which version of GALACTICUS the parameter file was constructed for. The migration script will then migrate the parameter file through all intermediate versions to bring it into compliance with the current version. You can also specify input and output versions directly:

```
scripts/aux/parametersMigrate.pl parameters.xml newParameters.xml --inputVersion 0.9.0 --outputVersion 0.9.3
```

will convert `parameters.xml` from version 0.9.0 syntax to version 0.9.3 syntax.

### 7.5.3. Computing Emission Lines

GALACTICUS can compute emission line luminosities for galaxies. This calculation is based on the methodology of Panuzzo et al. [2003]. Briefly, HII region models are constructed using CLOUDY for a variety of gas densities and metallicities, and HI, HeI, and OII ionizing luminosities. Emission line luminosities are then computed by interpolating in these tables based on the instantaneous properties of model galaxies.

To compute emission line luminosities it is therefore necessary to run GALACTICUS including the following rest-frame luminosity filters at each redshift of interest: Lyc, HeliumContinuum, OxygenContinuum. This causes the ionizing luminosity for each three species to be computed and output. Emission line luminosities can then be accessed as named properties using names such as \texttt{totalLineLuminosity:balmerAlpha6563:rest:0.0000}, which will return the H\textalpha luminosity at $z = 0$. Equivalent properties for disk and spheroid are provided (simply replace “total” with “disk” or “spheroid”). Currently available lines are:

- \texttt{balmerAlpha6563}
- \texttt{balmerBeta4861}
- \texttt{oxygenII3726}
- \texttt{oxygenII3729}
- \texttt{oxygenII4959}
- \texttt{oxygenII5007}
- \texttt{nitrogenII6584}
- \texttt{sulfurII6731}
- \texttt{sulfurII6716}

Additionally, if a line is requested as \texttt{totalLineLuminosity:balmerAlpha6563:<filterName>:rest:0.0000} then the line luminosity is computed under the provided filter, and the luminosity is returned in units of maggies for easy conversion to magnitudes.
7.5.4. Reionization Calculations

GALACTICUS can self-consistently solve for the evolution of the IGM as it becomes photoionized by light emitted by stars and AGN. To activate this calculation, include the following in your parameters file:

```xml
<!-- IGM evolver -->
<parameter>
  <name>intergalacticMediumStateMethod</name>
  <value>internal</value>
</parameter>
<parameter>
  <name>igmPropertiesCompute</name>
  <value>true</value>
</parameter>
<parameter>
  <name>igmPropertiesTimeCountPerDecade</name>
  <value>10</value>
</parameter>

<!-- Background radiation -->
<parameter>
  <name>backgroundRadiationCompute</name>
  <value>true</value>
</parameter>
<parameter>
  <name>radiationIntergalacticBackgroundMethod</name>
  <value>internal</value>
</parameter>
<parameter>
  <name>backgroundRadiationWavelengthCountPerDecade</name>
  <value>50</value>
</parameter>
<parameter>
  <name>backgroundRadiationTimeCountPerDecade</name>
  <value>10</value>
</parameter>

<!-- Halo accretion options -->
<parameter>
  <name>accretionHaloMethod</name>
  <value>naozBarkana2007</value>
</parameter>
```

The first block of parameters switches GALACTICUS to using an internal calculation for the state of the IGM, instructs it to solve for IGM properties as a function of time, and specifies that IGM properties should be updated 10 times per decade of cosmic time. Specifically, at each of these time intervals, solving of galaxy evolution is halted and the IGM evolved up to this time using the currently computed photoionizing background spectrum.

The second block of parameters activates an internal calculation of cosmic background radiation, in which the background is computed from the emissivities of model galaxies and AGN. The number of points at which to tabulate the background per decade of wavelength and cosmic time are specified.
Finally, the third block of parameters tells Galacticus to use the Naoz and Barkana [2007] prescription for computing gas accretion into halos from the IGM. This prescription uses the filtering mass to determine accretion rates, and will take the filtering mass from the internal IGM evolution calculation.

With these three sets of configurations, Galacticus will perform a self-consistent evolution of the IGM—in the sense that the IGM is ionized by photons emitted by model galaxies and AGN, while galaxy evolution is affected by the computed state of the model IGM. Note that, when run in this way, Galacticus needs to keep all merger trees in memory simultaneously (as they are run synchronously to allow the IGM properties to evolve alongside galaxy properties).

Once completed, data on the IGM and background radiation are written to the output file in the igmProperties and backgroundRadiation groups respectively.
8. Troubleshooting

This chapter contains guidance on solving various problems that you might encounter when running Galacticus.

8.1. Low CPU utilization with large numbers of output redshifts

If a Galacticus model is failing to make use of the majority of the available CPU cycles, and you are running a model with a large number of output redshifts the problem may be that I/O to disk is limiting the rate at which merger trees can be processed. I/O occurs through the HDF5 library which provides caching functionality. Therefore, this problem can often be mitigated by expanding the size of the HDF5 library’s cache. Galacticus allows you to do this using a set of input parameters:

[hdf5CacheElementsCount] HDF5 limits the number of objects that it will store in its cache. Increasing this number will allow more data to be cached and potentially make disk I/O more efficient. We have had good results by setting this number to some factor (e.g. 2) times the product of the number of output redshifts and the number of properties being output in each snapshot.

[hdf5CacheSizeBytes] HDF5 also limits the size of the cache in bytes. We have had good results setting this to a factor of a few above the product of [hdf5CacheElementsCount], the chunk size (see below) and the size of each output property (8 bytes).

[hdf5SieveBufferSize] HDF5 uses a sieve buffer to speed reading/writing of partial datasets. Increasing the buffer size (specified here in bytes) may improve I/O performance. We have had good results using a value of 512Kb.

[hdf5UseLatestFormat] Normally, HDF5 selects an internal file format to used based on maximum portability. If you set this option to true HDF5 will instead use the latest format that it supports—typically allowing it to employ optimizations unavailable to older versions of HDF5. Note that this can make the output file unreadable by older versions of the HDF5 library.

Additionally, you can ensure that compression is switched off in the HDF5 output by setting [hdf5CompressionLevel] = −1. Finally, adjusting the HDF5 chunk size via the [hdf5ChunkSize] parameter may make for more efficient I/O. HDF5 datasets are read/written in chunks of this size. Increasing the size may improve I/O performance.
9. Numerical Implementation

9.1. Timestepping Criteria

Galacticus evolves each merger tree by repeatedly walking the tree and evolving each node forward in time by some timestep $\Delta t$. Nodes are evolved individually such that nodes in different branches of a tree may have reached different cosmic times at any given point in the execution of Galacticus. Each node is evolved over the interval $\Delta t$ using an adaptive ordinary differential equation (ODE) solver, which adjusts the smaller timesteps, $\delta t$, taken in evolving the system of ODEs to maintain a specified precision.

The choice of $\Delta t$ then depends on other considerations. For example, a node should not be evolved beyond the time at which it is due to merge with another galaxy. Also, we typically don’t want satellite nodes to evolve too far ahead of their host node, such that any interactions between satellite and host occur (near) synchronously.

In the remainder of this section we list all criteria used to select $\Delta t$ for a node. All criteria are considered and the largest $\Delta t$ consistent with all criteria is selected.

9.1.1. Tree Criteria

The following timestep criteria ensure that tree evolution occurs in a way which correctly preserves tree structure and ordering of interactions between nodes.

“Branch Segment” Criteria

For nodes which are the primary progenitor of their parent, the “branch segment” criterion asserts that

$$\Delta t \leq t_{\text{parent}} - t$$  \hspace{1cm} (9.1)

where $t$ is current time in the node and $t_{\text{parent}}$ is the time of the parent node. This ensures that primary progenitor nodes to not evolve beyond the time at which their parent (which they will replace) exists. If this criterion is the limiting criteria for $\Delta t$ then the node will be promoted to replace its parent at the end of the timestep.

“Parent” Criteria

For nodes which are satellites in a hosting node the “parent” timestep criterion asserts that

$$\Delta t \leq t_{\text{host}},$$  \hspace{1cm} (9.2)

$$\Delta t \leq \epsilon_{\text{host}}(a/\dot{a}),$$  \hspace{1cm} (9.3)

where $t_{\text{host}} = \text{[timestepHostAbsolute]}$, $\epsilon_{\text{host}} = \text{[timestepHostRelative]}$, and $a$ is expansion factor. These criteria are intended to prevent a satellite for evolving too far ahead of the host node before the host is allowed to “catch up”.
“Satellite” Criteria

For nodes which host satellite nodes, the “satellite” criterion asserts that
\[ \Delta t \leq \min(t_{\text{satellite}}) - t, \] (9.4)
where \( t \) is the time of the host node and \( t_{\text{satellite}} \) are the times of all satellite nodes in the host. This criterion prevents a host from evolving ahead of any satellites.

“Sibling” Criteria

For nodes which are primary progenitors, the “sibling” criterion asserts that
\[ \Delta t \leq \min(t_{\text{sibling}}) - t, \] (9.5)
where \( t \) is the time of the host node and \( t_{\text{sibling}} \) are the times of all siblings of the node. This criterion prevents a node from reaching its parent (and being promoted to replace it) before all of its siblings have reach the parent and have become satellites within it.

“Mergee” Criteria

For nodes with mergee nodes, the “mergee” criterion asserts that
\[ \Delta t \leq \min(t_{\text{merge}}) - t, \] (9.6)
where \( t \) is the time of the host node and \( t_{\text{merge}} \) are the times at which the mergees will merge. This criterion prevents a node from evolving past the time at which a merger event takes place.

9.1.2. General Criteria

“Simple” Criteria

The “simple” timestep criteria assert that
\[ \Delta t \leq t_{\text{simple}}, \] (9.7)
\[ \Delta t \leq \epsilon_{\text{simple}}(a/\dot{a}), \] (9.8)
where \( t_{\text{simple}} = [\text{timestepSimpleAbsolute}] \), \( \epsilon_{\text{simple}} = [\text{timestepSimpleRelative}] \), and \( a \) is expansion factor. These criteria are intended to prevent any one node evolving over an excessively large time in one step. In general, these criteria are not necessary, as nodes should be free to evolve as far as possible unless prevented by some physical requirement. These criteria are therefore present to provide a simple example of how timestep criteria work.

“Satellite” Criteria

The “satellite” timestep criteria asserts the following for satellite nodes. If the satellite’s merge target has been advanced to at least a time of \( t_{\text{required}} = t_{\text{satellite}} + \Delta t_{\text{merge}} - \delta t_{\text{merge,maximum}} \) then
\[ \Delta t \leq \Delta t_{\text{merge}}, \] (9.9)
where \( t_{\text{satellite}} \) is the current time for the satellite node, \( \Delta t_{\text{merge}} \) is the time until the satellite is due to merge and \( \delta t_{\text{merge,maximum}} \) is the maximum allowed time difference between merging galaxies. This ensures that the satellite is not evolved past the time at which it is due to merge. If this criterion is the limiting criteria for \( \Delta t \) then the merging of the satellite will be triggered at the end of the timestep.
9.1. Timestepping Criteria

If the merge target has not been advanced to at least \( t_{\text{required}} \) then instead

\[
\Delta t \leq \max(\Delta t_{\text{merge}} - \delta t_{\text{merge,maximum}}/2, 0),
\]  

(9.10)

is asserted to ensure that the satellite does not reach the time of merging until its merge target is sufficiently close (within \( \delta t_{\text{merge,maximum}} \)) of the time of merging.

9.1.3. Output Criteria

“History” Criteria

The “history” timestep criterion asserts that

\[
\Delta t \leq t_{\text{history},i} - \bar{t}
\]  

(9.11)

where \( t \) is the current time, \( t_{\text{history},i} \) is the \( i \)th time at which the global history (see §5.1.5) of galaxies is to be output and \( i \) is chosen to be the smallest \( i \) such that \( t_{\text{history},i} > t \). If there is no \( i \) for which \( t_{\text{history},i} > t \) this criterion is not applied. If this criterion is the limiting criterion for \( \Delta t \) then the properties of the galaxy will be accumulated to the global history arrays at the end of the timestep.

“Main Branch Evolution” Criteria

If \( \text{timestepRecordEvolution} = \text{true} \), then the “main branch evolution” timestep criterion asserts that

\[
\Delta t \leq t_{\text{record},i} - \bar{t}
\]  

(9.12)

where \( t \) is the current time, \( t_{\text{record},i} \) is the \( i \)th time at which the evolution of main branch galaxies is to be output and \( i \) is chosen to be the smallest \( i \) such that \( t_{\text{record},i} > t \). If there is no \( i \) for which \( t_{\text{record},i} > t \) this criterion is not applied. If this criterion is the limiting criterion for \( \Delta t \) then the properties of the galaxy will be recorded at the end of the timestep.
Part II.

Advanced Use
10. Constraining \textsc{Galacticus}

10.1. Model Accuracy

The model accuracy script processes the model described by a standard \textsc{Galacticus} constraints configuration file to assess how accurate the model is. Specifically, it determines the relative contribution to the covariance of each constraint arising from the finite number of merger trees run in the model and the intrinsic covariance of the observations. An accurate model should make a negligible contribution to the covariance.

To run the model accuracy script use

```
constraints/testModelAccuracy.pl <configFile>
```

where `configFile` is the name of the configuration file. The script will run the model multiple times, reducing the number of trees per decade by a factor of two each time (this is done 8 times, such that the smallest model run has a factor 128 times fewer trees than the original model). Furthermore, this sequence of models is run for three different choices for sampling halo masses: `powerLaw`, `haloMassFunction`, and `stellarMassFunction` (see §13.8.6).

For each constraint in the specified constraint compilation and accuracy measure is constructed which is the root mean squared ratio of the error arising from the finite number of trees and the intrinsic error of the constraint data.

Accuracy analysis files are written to:

```
<workDirectory>/accuracy
```

For each constraint in the compilation a plot showing accuracy measure as a function of CPU time is written to:

```
<constraintLabel>_mergerTreeBuildTreesPerDecade.pdf
```

Each point in the plot is labelled with the number of merger trees per decade. An accurate model should have accuracy measure significantly below unity. This plot is also useful to see which sampling method achieves that accuracy in the least amount of CPU time.

Additionally, a report is written to:

```
<constraintLabel>Report.txt
```

This file lists, for each sampling method, the accuracy measure achieved and the CPU time taken for the largest model run.

10.2. Model Convergence

The model convergence script processes the model described by a standard \textsc{Galacticus} constraints configuration file to assess how well converged the model is with respect to several of \textsc{Galacticus}'s numerical parameters. Specifically, it determines the a covariance measure for each constraint in the specified compilation file.

To run the model convergence script use
constraints/testConvergence.pl <configFile>

where **configFile** is the name of the configuration file. The script will run the model multiple times, adjusting the value of a numerical parameter each time.

For each constraint in the specified constraint compilation a convergence measure, \( C \), is constructed which

\[
C = \sum_i \frac{(y_i - y_{\text{ideal},i})^2}{2\sigma_{\text{ideal},i}^2}
\]  

(10.1)

where \( y_i \) is the result of the constraint, \( \sigma_i \) is the error on the result, and subscript “ideal” refers to the model with the most ideal value (i.e. that in the original, unmodified model) of the numerical parameter being tested (which might be the lowest value for a mass resolution, or the highest value for the maximum tree mass simulated for example). To be converged, the convergence measure should remain consistent with unity within a significant distance\(^1\) away from the ideal value.

Convergence analysis files are written to:

<workDirectory>/convergence

For each combination of constraint in the compilation and numerical parameter a plot showing the convergence measure as a function of the numerical parameter is created in:

<constraintLabel>_<numericalParameter>.pdf

Each point in the plot has an error bar since, due to the limited number of merger trees run, the convergence measure is not known with perfect precision. A horizontal line shows the desired convergence measure of unity.

Additionally, a report is written to:

<constraintLabel>Report.txt

This file lists, for each numerical parameter, the convergence measure and its error achieved by the ideal model. Additionally a normalized measure (the measure divided by its error) is listed. This normalized measure can be approximately interpreted as the number of \( \sigma \) deviation from convergence.

### 10.3. Model Discrepancy

Model discrepancy scripts process the model described by a standard GALACTICUS configuration file to produce an output HDF5 file which describes a particular contribution to the model discrepancy. The format of these files is

HDF5 "discrepancy.hdf5" {
GROUP "/" {
  DATASET "additive" {
  }
  DATASET "multiplicative" {
  }
  DATASET "covariance" {
  }
}
}

\(^1\)“Significant distance” here requires some judgement. Typically we would like for the model results to not change significantly as the value of a numerical parameter is adjusted by at least a factor of 2 away from the ideal value.
Each of the three datasets is optional (i.e. not all need be provided for each discrepancy). The *additive* dataset gives an additive offset which will be applied to the relevant model results. The *multiplicative* dataset similarly gives a multiplicative offset which will be applied to the relevant model results. Finally, the *covariance* dataset gives the contribution from this discrepancy to the covariance matrix used in evaluating the model likelihood.

Discrepancy files are written to

\<workDirectory\>/modelDiscrepancy/<discrepancyLabel>/discrepancy<constraintLabel>.hdf5

Constraint scripts (see §10.5) accept a command line option `–modelDiscrepancies` which specifies the path to the `modelDiscrepancy` directory (i.e. `<workDirectory>/modelDiscrepancy`) and will search for any relevant model discrepancy files and apply them in their calculations.

### 10.3.1. Monte Carlo Merger Trees

*Galacticus* typically uses Monte Carlo-generated merger trees when being constrained to fit data. These have the advantage that they can be generated for any cosmological parameters (necessary if the cosmological parameters are to be varied as part of the constraining process) and they can be generated uniquely for each model evaluation which avoids any bias introduced by using a fixed set of halos.

However, these Monte Carlo-generated trees may not precisely capture the properties of merger trees derived from a fully non-linear calculation of gravitational collapse (e.g. as performed by an N-body simulation). Therefore it is important to assess the model discrepancy arising from this limitation.

Model discrepancy files can be generated using:

```
constraints/modelDiscrepancy/monteCarloTrees.pl config.xml
```

where `config.xml` is a standard *Galacticus* constraint configuration file. The script will run two sets of models, one using N-body merger trees derived from the Millennium Simulation, and a second using Monte Carlo-generated merger trees. The number of subvolumes of the Millennium Simulation to use is specified by the `subVolumeCount` option to this script (a default of 32 subvolumes is used if no number is specified). The subvolume data will be downloaded from the Millennium Simulation database if necessary.

To make a fair comparison, Millennium Simulation merger trees have their branches pruned below a mass corresponding to $20$ particles, and the Monte Carlo merger trees are built with the equivalent mass resolution. Additionally, the Monte Carlo merger trees are regridded onto a set of timesteps matched to the Millennium Simulation.

A multiplicative model discrepancy is computed for each constraint included in the compilation (as specified in the configuration file) equal to the ratio of the N-body result to the Monte Carlo result. Additionally, the subvolumes of the Millennium Simulation are used to estimate the covariance in the N-body result due to the finite volume of the simulation. The result is computed for each subvolume separately and the covariance of the result between subvolumes computed. This is repeated using pairs of subvolumes, quads of subvolumes, etc. If $2^n$ subvolumes were used, then the covariance measured from the result combining $2^{n-3}$ subvolumes is used to extrapolate the covariance for all 512 subvolumes assuming that the covariance scales in inverse proportion to the number of subvolume used. Finally, the contribution of the Monte Carlo trees model to the covariance is assumed to be a diagonal matrix with elements equal to the square of the reported errors on the result of the model.

### 10.3.2. Fixed Virial Orbits

The orbital parameters of subhalos at the point of virial orbit crossing are usually drawn from an appropriate cosmological distribution. If instead fixed virial orbital parameters are used instead then term should be included in the model discrepancy accounting for this approximation.

Model discrepancy files can be generated using:
10. Constraining Galacticus

```
constraints/modelDiscrepancy/fixedVirialOrbits.pl config.xml
```

where `config.xml` is a standard Galacticus constraint configuration file. The script will run two models, one using fixed virial orbital parameters, and a second using variable orbital parameters using the Benson2005 method (see §13.44.2). A multiplicative model discrepancy is computed for each constraint included in the compilation (as specified in the configuration file) equal to the ratio of the variable orbits result to the fixed orbits result. Additionally, a model discrepancy covariance is computed. This is assumed to be a diagonal matrix with elements equal to the square of the reported errors on the results of the fixed and variable orbital parameters models.

10.3.3. Jiang et al. (2008) Merger Time Scatter

The Jiang et al. [2008] algorithm for the merging times of dark matter subhalos includes drawing times from a log-normal distribution of width $\sigma = 0.4$ with median equal to their fitting function (see §13.44.1). If instead zero scatter is used then a term should be included in the model discrepancy accounting for this approximation.

Model discrepancy files can be generated using:

```
constraints/modelDiscrepancy/jiang2008MergingTimeScatter.pl config.xml
```

where `config.xml` is a standard Galacticus constraint configuration file. The script will run two models, one using the default scatter specified by the configuration file, and a second using $\sigma = 0.4$. A multiplicative model discrepancy is computed for each constraint included in the compilation (as specified in the configuration file) equal to the ratio of the $\sigma = 0.4$ and default scatter results. Additionally, a model discrepancy covariance is computed. This is assumed to be a diagonal matrix with elements equal to the sum of the square of the reported errors on the result of the default scatter and $\sigma = 0.4$ models.

10.4. Optimal Halo Mass Function Sampling

Suppose we want to fit parameters of the Galacticus model to some dataset. The basic approach is to generate large numbers of model realizations for different parameter values and see which ones best match the data. Galacticus models involve simulating individual merger trees and then adding together their galaxies to produce some overall function. The question we want to answer is, given some finite amount of computing time, what is the optimal distribution of halo masses to run when comparing to a given dataset. For example, is it better to run a volume limited sample (as one would get from an N-body simulation) or is it better to use, say, equal numbers of halos per logarithmic interval of halo mass? The following section describes how to solve this optimization problem in the specific case of fitting to the stellar mass function.

10.4.1. Li & White (2009) Stellar Mass Function

First, some definitions:

- $n(M)d\ln M$ is the dark matter halo mass function, i.e. the number of halos in the range $M$ to $M + Md\ln M$ per unit volume;
- $\gamma(M)d\ln M$ is the number of trees that we will simulate in the range $M$ to $M + Md\ln M$;
- $\alpha(M_*)$ is the error on the observed stellar mass function at mass $M_*$;
- $P(N|M_*, M; \delta\ln M_*)$ is the conditional stellar mass distribution function of galaxies of stellar mass $M_*$ in a bin of width $\delta\ln M_*$ per halo of mass $M$;
10.4. Optimal Halo Mass Function Sampling

$t(M)$ is the CPU time it takes to simulate a tree of mass $M$.

To clarify, $P(N|M_*, \delta \ln M_*; \delta \ln M_*)$ is the probability to find $N$ galaxies of mass between $M_*$ in a bin of width $\delta \ln M_*$ in a halo of mass $M$. The usual conditional stellar mass function is simply the first moment of this distribution:

$$\phi(M_*; M) \delta \ln M_* = \sum_{N=0}^\infty N P(N|M_*, M; \delta \ln M_*)$$  \hspace{1cm} (10.2)

The model estimate of the stellar mass function $\Phi(M_*)$ (defined per unit $\ln M_*$) is

$$\Phi(M_*) = \int_0^\infty \phi(M_*; M) \frac{n(M)}{\gamma(M)} \gamma(M) d \ln M,$$  \hspace{1cm} (10.3)

where the $n(M)/\gamma(M)$ term is the weight assigned to each tree realization—and therefore the weight assigned to each model galaxy when summing over a model realization to construct the stellar mass function.

When computing a model likelihood, we must employ some statistic which defines how likely the model is given the data. Typically, for stellar mass functions we have an estimate of the variance in the data, $\alpha^2(M_*)$, as a function of stellar mass (full covariance matrices are typically not provided but, ideally would be, and can be easily incorporated into this method). In that case, we can define a likelihood

$$\ln L = -\frac{1}{2} \sum_i \frac{[\phi_{\text{obs},i} - \phi_i]^2}{\alpha^2_i + \sigma^2_i}$$  \hspace{1cm} (10.4)

where the sum is taken over all data points, $i$, and $\sigma^2_i$ is the variance in the model estimate and is given by

$$\sigma^2(M_*) = \langle [\phi(M_*) - \bar{\phi}(M_*)]^2 \rangle,$$  \hspace{1cm} (10.5)

where $\phi(M_*)$ is the realization from a single model and $\bar{\phi}(M_*)$ is the model expectation from an infinite number of merger tree realizations and the average is taken over all possible model realizations. Since the contributions from each merger tree are independent,

$$\sigma^2(M_*) = \sum_i \zeta^2_i(M_*; M)$$  \hspace{1cm} (10.6)

where $\zeta^2_i(M_*; M)$ is the variance in the contribution to the stellar mass function from tree $i$. This in turn is given by

$$\zeta^2_i(M_*; M) = \psi^2(M_*; M) \left[ \frac{n(M)}{\gamma(M)} \right]^2,$$  \hspace{1cm} (10.7)

where $\psi^2(M_*; M)$ is the variance in the conditional stellar mass function. In the continuum limit this becomes

$$\sigma^2(M_*) = \int_0^\infty \psi^2(M_*; M) \left[ \frac{n(M)}{\gamma(M)} \right]^2 \gamma(M) d \ln M.$$  \hspace{1cm} (10.8)

Model variance artificially increases the likelihood of a given model. We would therefore like to minimize the increase in the likelihood due to the model variance:

$$\Delta 2 \ln L = \sum_i \frac{[\phi_{\text{obs},i} - \phi_i]^2}{\alpha^2_i} - \frac{[\phi_{\text{obs},i} - \phi_i]^2}{\alpha^2_i + \sigma^2_i}$$  \hspace{1cm} (10.9)

\(^2\)To put it another way, $P(N|M_*, M; \delta \ln M_*)$ is closely related to the commonly used Halo Occupation Distribution.
Of course, we don’t know the model prediction, \( \phi_i \), in advance\(^3\). However, if we assume that a model exists which is a good fit to the data then we would expect that \( [\phi_{\text{obs}, i} - \phi_i]^2 \approx \alpha_i^2 \) on average. In that case, the increase in likelihood due to the model is minimized by minimizing the function\(^4\)

\[
F[\gamma(M)] = \sum_i \frac{\alpha_i^2}{\alpha_i^2 + \sigma_i^2}.
\]  

(10.10)

If the bins all have the same \( \delta \ln M_* \) we can turn the sum into an integral

\[
F[\gamma(M)] = \int_0^\infty \frac{\alpha(M_*)^2}{\alpha(M_*)^2 + \sigma(M_*)^2} \, d \ln M_*.
\]  

(10.11)

Obviously, the answer is to make \( \gamma(M) = \infty \), in which case \( F[\gamma(M)] = 0 \). However, we have finite computing resources. The total time to run our calculation is

\[
\tau = \int_0^\infty t(M) \gamma(M) \, d \ln M.
\]  

(10.12)

We therefore want to minimize \( F[\gamma(M)] \) while keeping \( \tau \) equal to some finite value. We can do this using a Lagrange multiplier and minimizing the function

\[
F[\gamma(M)] = \int_0^\infty \frac{\alpha(M_*)^2}{\alpha(M_*)^2 + \sigma(M_*)^2} \, d \ln M_* + \int_0^\infty \lambda \gamma(M) t(M) \, d \ln M.
\]  

(10.13)

Finding the functional derivative and setting it equal to zero gives:

\[
\gamma(M) = \sqrt{\frac{\xi(M)}{\lambda t(M)}},
\]  

(10.14)

in the limit where\(^5\) \( \sigma(M_*) \ll \alpha(M_*) \), and where

\[
\xi(M) = n^2(M) \int_{-\infty}^\infty \frac{\psi^2(M_*; M)}{\alpha^2(M_*)} \, d \ln M_*. \]  

(10.15)

The values of \( \lambda \) and \( \delta \ln M_* \), and the normalization of \( t(M) \) are unimportant here since we merely want to find the optimal shape of the \( \gamma(M) \) function—we can then scale it up or down to use the available time.

Figure 10.2 shows the function \( \gamma(M) \) obtained by adopting a model conditional stellar mass function which is a sum of central and satellite terms. Specifically, we use the model of Leauthaud et al. \([2011]\) which is constrained to match observations from the COSMOS survey. In their model\(^6\):

\[
\langle N_c(M_*|M) \rangle \equiv \int_{M_*}^{\infty} \phi_c(M'_*|d \ln M'_* = \frac{1}{2} \left[ 1 - \text{erf} \left( \frac{\log_{10} M_* - \log_{10} f_{SHMR}(M)}{\sqrt{2} \sigma_{\log M_*}} \right) \right].
\]  

(10.16)

Here, the function \( f_{SHMR}(M) \) is the solution of

\[
\log_{10} M = \log_{10} M_1 + \beta \log_{10} \left( \frac{M_*}{M_{*,0}} \right) + \frac{(M_* / M_{*,0})^\delta}{1 + (M_* / M_{*,0})^{-\gamma}} - 1/2.
\]  

(10.17)

\(^3\)Below, we will adopt a simple empirical model for \( \phi(M_*) \). However, it should not be used here since we will in actuality be computing the likelihood from the model itself.

\(^4\)This can be seen intuitively: we are simply requiring that the variance in the model prediction is small compared the the variance in the data.

\(^5\)This is the limit in which we would like our results to be.

\(^6\)This integral form of the conditional stellar mass function is convenient here since it allows for easy calculation of the number of galaxies expected in the finite-width bins of the observed stellar mass function.
Table 10.1.: Parameters of the conditional stellar mass function fit.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha_{\text{sat}} )</td>
<td>1.0</td>
</tr>
<tr>
<td>( \log_{10} M_1 )</td>
<td>12.120</td>
</tr>
<tr>
<td>( \log_{10} M_{*,0} )</td>
<td>10.516</td>
</tr>
<tr>
<td>( \beta )</td>
<td>0.430</td>
</tr>
<tr>
<td>( \delta )</td>
<td>0.5666</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>1.53</td>
</tr>
<tr>
<td>( \sigma_{\log M_*} )</td>
<td>0.206</td>
</tr>
<tr>
<td>( B_{\text{cut}} )</td>
<td>0.744</td>
</tr>
<tr>
<td>( B_{\text{sat}} )</td>
<td>8.00</td>
</tr>
<tr>
<td>( \beta_{\text{cut}} )</td>
<td>-0.13</td>
</tr>
<tr>
<td>( \beta_{\text{sat}} )</td>
<td>0.859</td>
</tr>
</tbody>
</table>

For satellites,

\[
\langle N_s(M_*, M) \rangle \equiv \int_{M_*}^{\infty} \phi_s(M_*) d \ln M_* = \langle N_s(M_*, M) \rangle \left( \frac{f_{\text{SHMR}}^{-1}(M_*)}{M_{\text{sat}}} \right)^{\alpha_{\text{sat}}} \exp \left( -\frac{M_{\text{cut}}}{f_{\text{SHMR}}^{-1}(M_*)} \right),
\]  

(10.18)

where

\[
\frac{M_{\text{sat}}}{10^{12} M_{\odot}} = B_{\text{sat}} \left( \frac{f_{\text{SHMR}}^{-1}(M_*)}{10^{12} M_{\odot}} \right)^{\beta_{\text{sat}}},
\]  

(10.19)

and

\[
\frac{M_{\text{cut}}}{10^{12} M_{\odot}} = B_{\text{cut}} \left( \frac{f_{\text{SHMR}}^{-1}(M_*)}{10^{12} M_{\odot}} \right)^{\beta_{\text{cut}}},
\]  

(10.20)

We use the best fit parameters from the SIG_MOD1 method of Leauthaud et al. [2011] for their \( z_1 \) sample, but apply a shift of \(-0.2\) dex in masses to bring the fit into line with the \( z = 0.07 \) mass function of Li and White [2009]. The resulting parameter values are shown in Table 10.1.

We assume that \( P_s(N|M_*, M; \Delta \ln M_*) \) is a Poisson distribution while \( P_c(N|M_*, M; \Delta \ln M_*) \) has a Bernoulli distribution, with each distribution’s free parameter fixed by the constraint of eqn. (10.2), and the assumed forms for \( \phi_c \) and \( \phi_s \).

The errors in the Li and White [2009] observed stellar mass function are well fit by (see Fig. 10.1):

\[
\alpha(M_*) = 10^{-3} \left( \frac{M_*}{4.5 \times 10^{10} M_{\odot}} \right)^{-0.3} \exp \left( -\frac{M_*}{4.5 \times 10^{10} M_{\odot}} \right) + 10^{-7},
\]  

(10.21)

and the tree processing time in GALACTICUS can be described by:

\[
\log_{10} t(M) = \sum_{i=0}^{2} C_i [\log_{10} M]^i
\]  

(10.22)

with \( C_0 = -0.73, \) \( C_1 = -0.20 \) and \( C_2 = 0.035. \)

The resulting optimal sampling density curve is shown in Fig. 10.2 and is compared to weighting by the halo mass function (i.e. the result of sampling halos at random from a representative volume). Optimal sampling gives less weight to low mass halos (since a sufficient accuracy can be obtained without the need to run many tens of thousands of such halos) and to high mass halos which are computationally expensive.
Errors on the stellar mass function at $z = 0.1$

Figure 10.1.: Errors on the Li and White [2009] stellar mass function (points) and the fitting function (line) given by eqn. (10.21).
Figure 10.2.: Optimal weighting (yellow line) compared with weighting by the dark matter halo mass function (i.e. sampling halos at random from a representative volume; blue line). Sampling densities have been normalized to unit compute time.
10. Constraining GALACTICUS

10.4.2. Refining by Other Merger Tree Statistics

Since building merger trees is relatively fast, while solving the baryonic physics is slow it may be advantageous to non-uniformly sample the distribution of merger trees at fixed merger tree mass, \( M \). For example, we could assign some measure of formation history to each merger tree, such as the time since the last major merger, \( \tau \). The halo mass function then becomes \( n(M, \tau) \) (which can be computed by simulating large numbers of trees), and the tree sampling function becomes \( \gamma(M, \tau) \). We’d then need to know the stellar mass function conditioned on both \( M \) and \( \tau \), \( \phi_*(M_*|M, \tau) \). Given these, the above approach could be easily generalized to determine an optimal \( \gamma(M, \tau) \). Then, after generating a merger tree, we’d first compute \( \tau \). If a sufficient number of trees in that \( \tau \) interval had already been computed, then we’d simply drop that tree and compute another one. The speed up here would depend on how fast building trees is relative to solving baryonic physics and what fraction of trees you discard. In principle, the trees could be generated, sampled and stored in advance so that we’d already have an optimally distributed set of trees in \( M \) and \( \tau \) that could be used for each model run.

10.5. Constraints

Any constraint which can be applied to GALACTICUS is defined by two files, a configuration file and a likelihood script, which must be placed in constraints/constraints and constraints/scripts respectively.

10.5.1. Configuration File

The configuration file should have the form:

```xml
<!-- Defines a constraint to match some data. -->
<constraint>
  <name>Long-form name of this constraint</name>
  <label>shortLabelForThisConstraint</label>
  <outputRedshift>0.07</outputRedshift>
  <outputRedshift>1.00</outputRedshift>
  <haloMassResolution>5.0e9</haloMassResolution>
  <haloMassMinimum>2.0e10</haloMassMinimum>
  <haloMassMaximum>2.0e14</haloMassMaximum>
  <analysis>constraints/scripts/myAnalysisScript.pl</analysis>
  <luminosity>
    <filter>UKIRT_K</filter>
    <redshift>0.0</redshift>
    <frame>rest</frame>
  </luminosity>
  <luminosity>
    <filter>UKIRT_K</filter>
    <redshift>1.0</redshift>
    <frame>observed</frame>
  </luminosity>
  <optionOn>outputMainBranchStatus</optionOn>
  <optionOn>outputDensityContrastData</optionOn>
  <parameter>
    <name>outputDensityContrastValues</name>
    <value>200.0</value>
  </parameter>
</constraint>
```
10.5. Constraints

The name and label are used to describe the constraint (label is used as a suffix in file names so should not contain spaces or other characters which might cause problems in file names).

The remaining elements describe the requirements for this constraint. haloMassResolution specifies the maximum resolution in mergers trees that still allows this constraint to be computed accurately. Similarly, haloMassMinimum and haloMassMaximum specify the required range of halo masses to simulate to allow this constraint to be computed accurately.

One or more outputRedshift elements may be present, each specifying a redshift at which output is required for this constraint. Similarly, one or more luminosity elements may be present, each of which specifies a luminosity which must be computed for this constraint. Each luminosity must contain a specification of filter, redshift, and frame to define which luminosity is to be computed.

One or more optionOn elements may be present. Each element must specify the name of a Galacticus input parameter. That parameter will be set to true in the Galacticus input parameter file.

Finally, arbitrary other parameter may be set using the standard parameter element which should give the name and value for the parameter. Optionally, an accumulation element may also be specified for each parameter. This controls how values of the parameter are to be accumulated if set by more than one constraint. An accumulation of overwrite will simply overwrite any previously set values. An accumulation of combine will concatenate all values set by different constraints. Finally, an accumulation of unique will concatenate all values set by different constraints and then filter out any duplicates.

When multiple constraints are used, their requirements are automatically combined.

10.5.2. Likelihood Script

The likelihood script for a constraint is required to perform several tasks, controlled by command line options. The script should accept the following command line syntax:

myScript.pl <galacticusFile> [options...]

where galacticusFile is the file name of the Galacticus model for which the likelihood calculation should be performed. The following options must be supported by the script:

-plotFile <fileName> If this option is present, the script should generate a plot showing the constraint and the model result and write it to fileName.

-outputFile <fileName> If this option is present, the script should compute the log-likelihood of the model given the constraint and write it to fileName using the format

<constraint>
<logLikelihood>-123</logLikelihood>
</constraint>

-accuracyFile <fileName> If this option is present, the script should write an XML file giving details of the accuracy of the model results relative to the observational errors using the format

<accuracy>
<x>...</x>
.
.
.
</accuracy>
10. Constraining Galacticus

In this file the \texttt{yModel} and \texttt{yData} elements should give the values of the model result and the comparable data respectively, while \texttt{errorModel} and \texttt{errorData} should give an estimate of the errors on these quantities. In the case of the model error this should include only the contribution arising from the finite number of merger trees simulated. This file will be used to judge whether the model is running sufficient merger trees such that the likelihood is not dominated by these errors. The \texttt{x} elements are optional but can be used to give the parameter values associated with each model result.

\texttt{-resultFile <fileName>} If this option is present, the script should write an XML file giving details of the result of the model using the format

\begin{verbatim}
<accuracy>
  <x>...</x>
  ...
  ...
  <x>...</x>
  <y>...</y>
  ...
  ...
  <y>...</y>
  <error>...</error>
  ...
  ...
  <error>...</error>
</accuracy>
\end{verbatim}
10.5. Constraints

In this file the $y$ elements should give the values of the model result, while the $error$ elements should give an estimate of the errors on these results. The error should include only the contribution arising from the finite number of merger trees simulated. This file will be used to judge whether the model result is converged with respect to various numerical parameters in GALACTICUS. The $x$ elements are optional but can be used to give the parameter values associated with each model result.

`--modelDiscrepancies <path>` If this option is present, the script should scan $\text{path}$. For each directory found in $\text{path}$ the script should check for the existence of a file named $\text{discrepancy<label>.hdf5}$ where $\text{label}$ is the label given for this constraint in its configuration file (see §10.5.1). If present, the model discrepancy given in that file should be applied to the likelihood calculation. See §10.3 for a description of the structure of the discrepancy files.

10.5.3. Available Constraints

**Li & White (2009) SDSS Stellar Mass Function**

This constraint utilizes the stellar mass function for $z \approx 0.07$ galaxies measured by Li and White [2009] from the Sloan Digital Sky Survey (SDSS). The mass function reported by Li and White [2009] is converted to the appropriate Hubble constant for the given GALACTICUS model (assuming that masses scale as $H_0^{-2}$ and volumes as $H_0^3$)—no adjustment is made for cosmological parameters given the low redshift of the sample.

Given a GALACTICUS model, total stellar masses of model galaxies are adjusted using:

$$M_* \rightarrow G M_*$$

where the $S$ operator is a multiplicative factor accounting for systematic errors in stellar mass determination and is equal to [Behroozi et al., 2010]

$$\log_{10} S = \mu + \kappa \log_{10} \left( \frac{M_*}{10^{11.3} M_\odot} \right)$$

where $\mu = [sdssStellarMassFunctionZ0.07StellarMassSystematicMu]$, $\kappa = [sdssStellarMassFunctionZ0.07StellarMassSystematicKappa]$, and the $G$ operator is a multiplicative factor drawn from a log-normal distribution of width 0.07 dex for each galaxy to mimic the effects of random errors on stellar masses (motivated by the discussion of Behroozi et al. [2010]).

The model masses are then used to construct a mass function by binning into a histogram using the masses reported by Li and White [2009] (modified as described above) as the centers of the bins (with bin boundaries placed at the geometric means of consecutive bin centers).

If the `--modelDiscrepancies` option is given, then any multiplicative or additive discrepancies found are applied to the model mass function, and any additional covariance is added to the covariance matrix.

The covariance matrix is computed as

$$C = C_{\text{obs}} + C_{\text{model,random}} + \sum_i C_{\text{discrepancy},i},$$

where $C_{\text{obs}}$ is the covariance matrix of the observational data, $C_{\text{model,random}}$ is the covariance matrix of the model arising from random noise (due to the finite number of trees simulated—see §10.5.3 for a description of how this covariance matrix is estimated), and $C_{\text{discrepancy},i}$ is the covariance due to the $i^{th}$ model discrepancy.

The model likelihood is then computed using:

$$L = \frac{1}{\sqrt{(2\pi)^n|C|}} \exp \left( -\frac{1}{2} \Delta C^{-1} \Delta \right),$$
where $\Delta_i = \Phi_{\text{model},i} - \Phi_{\text{observed},i}$ is the difference between the model and observed mass functions, and $n$ is the number of points in the mass function histogram.

Computing the large-scale structure contribution to the covariance function requires integration of the non-linear matter power spectrum over the Fourier transform of the survey window function. We use the method of Peacock and Dodds [1996] to determine the non-linear matter power spectrum, because of its simplicity and speed. We have checked that using a more accurate non-linear matter power spectrum (e.g. Lawrence et al. 2010) makes negligible difference to our results.

To find a suitable halo occupation distribution (HOD) to describe the galaxies in the Li and White [2009] sample we adopt the model of Behroozi et al. [2010]. This is an 11 parameter model which describes separately the numbers of satellite and central galaxies occupying a halo of given mass—the reader is referred to Behroozi et al. [2010] for a complete description of the functional form of this parametric HOD.

To reproduce the mass function of Li and White [2009] using this HOD we use the Bayesian inference engine (BIE) [Weinberg, 2012] to constrain the HOD parameters. We use a likelihood

$$\ln \mathcal{L} = -\frac{1}{2} \Delta \cdot \mathcal{C}^{-1} \cdot \Delta^T - \frac{N}{2} \ln(2\pi) - \frac{\ln |\mathcal{C}|}{2},$$

(10.27)

where $N$ is the number of bins in the mass function, $\mathcal{C}$ is the covariance matrix of the observed mass function, and $\Delta_i = \phi_i^{(\text{HOD})} - \phi_i^{(\text{observed})}$. Of course, it is precisely this covariance matrix, $\mathcal{C}$, that we are trying to compute. We therefore adopt an iterative approach as follows:

1. make an initial estimate of the covariance matrix, assuming that only Poisson errors contribute (the covariance matrix is therefore diagonal, and the terms are easily computed from the measured mass function and the survey volume as a function of stellar mass);
2. find the maximum likelihood parameters of the HOD given the observed mass function and the current estimate of the covariance matrix;
3. using this HOD and the framework of Smith [2012], compute a new estimate of the covariance matrix, including all three contributions;
4. repeat steps 2 and 3 until convergence in the covariance matrix is achieved.

In practice we find that this procedure leads to an HOD and covariance matrix which oscillate between two states in successive iterations. The differences in the covariance matrix are relatively small however, so we choose to conservatively adopt the covariance matrix with the larger values. In future, adding additional constraints to the HOD (as described below) should help mitigate this problem.

**Martin et al. (2010) ALFALFA HI Mass Function**

This constraint utilizes the HI mass function for $z \approx 0.0$ galaxies measured by Martin et al. [2010] from the ALFALFA survey. The mass function reported by Martin et al. [2010] is converted to the appropriate Hubble constant for the given GALACTICUS model (assuming that masses scale as $H_0^{-2}$ and volumes as $H_0^3$)—no adjustment is made for cosmological parameters given the low redshift of the sample.

Given a GALACTICUS model, total gas masses of model galaxies are adjusted using:

$$M_{\text{HI}} \rightarrow GS M_{\text{gas}}$$

(10.28)

where the $S$ operator is a multiplicative factor accounting for systematic errors in HI mass determination and for the unknown molecular fraction and is equal to:

$$\log_{10} S = \mu + \kappa \log_{10} \left( \frac{M_*}{10^9 M_\odot} \right)$$

(10.29)
10.5. Constraints

Figure 10.3: The observational random error in galaxy HI mass as a function of HI mass for the ALFALFA survey. Points show the errors reported by Haynes et al. [2011], while the line shows a simple functional form fit to these errors.

where $\mu = \text{[alfalfaHiMassFunctionZ0.00MolecularFractionMu]}$, $\kappa = \text{[alfalfaHiMassFunctionZ0.00MolecularFractionKappa]}$, and the $G$ operator is a multiplicative factor drawn from a log-normal distribution. The width of this log-normal is determined from the combination of observational random errors on HI mass and scatter in the $H_2$/HI mass ratio at fixed total gas mass. Observational random errors on HI mass are taken from Fig. 19 of Haynes et al. [2011]. We fit the magnitude of the error as a function of HI mass using a functional form:

$$
\sigma_{\text{obs}} = a + \exp \left( -\frac{\log_{10}(M_{HI}/M_\odot) - b}{c} \right),
$$

(10.30)

where $\sigma_{\text{obs}}$ is the error on $\log_{10}(M_{HI}/M_\odot)$. We find a good fit using values of $a = 0.100$, $b = 5.885$, and $c = 0.505$ as shown in Fig. 10.5.3.

In addition, we expect there to be significant scatter in the $H_2$/HI mass ratio at fixed total gas mass. For example, Figure 5 of Power et al. [2010] shows a broad distribution of such values. We approximate this scatter as a Gaussian random process with standard deviation $\sigma$. This random “error” is added in quadrature to the observational errors when constructing the mass function. For a prior on $\sigma$ we adopt a normal distribution with mean of 0.4 (estimated from Figure 5 of Power et al. [2010]) and standard deviation 0.3.

The model masses are then used to construct a mass function by binning into a histogram using the masses reported by Martin et al. [2010] (modified as described above) as the centers of the bins (with bin boundaries placed at the geometric means of consecutive bin centers).

If the -modelDiscrepancies option is given, then any multiplicative or additive discrepancies found are applied to the model mass function, and any additional covariance is added to the covariance matrix.

The covariance matrix is computed as

$$
C = C_{\text{obs}} + C_{\text{model,random}} + \sum_i C_{\text{discrepancy},i},
$$

(10.31)

where $C_{\text{obs}}$ is the covariance matrix of the observational data, $C_{\text{model,random}}$ is the covariance matrix of the model arising from random noise (due to the finite number of trees simulated, and $C_{\text{discrepancy},i}$ is the covariance due to the $i^{th}$ model discrepancy.
To construct $C_{\text{model, random}}$ we make use of the fact that Galacticus works by sampling a set of tree “root masses” from the $z = 0$ dark matter halo mass function. From each root, a tree is grown, within which the physics of galaxy formation is then solved. Root masses are sampled uniformly from the halo mass function. That is, the cumulative halo mass function, $N(M)$, is constructed between the maximum and minimum halo masses to be simulated. The number of root masses, $N_r$, to be used in a model evaluation is then determined. Root masses are then chosen such that

$$N(M_i) = N(M_{\text{min}}) \frac{i - 1}{N_r - 1}$$

for $i = 1 \ldots N_r$ (noting that $N(M_{\text{max}}) = 0$ by construction).

Consider first those galaxies which form in the main branch of each tree (i.e. those galaxies which are destined to become the central galaxy of the $z = 0$ halo). Suppose that we simulate $N_k$ halos of root mass $M_k$ at $z = 0$. In such halos the main branch galaxies will, at any time, have stellar masses drawn from some distribution $p_k(M_\star | t)$. The number of such galaxies contributing to bin $i$ of the mass function is therefore binomially distributed with success probability $p_{ki} = \int_{M_{\text{min}}}^{M_{\text{max}}} p_k(M_\star | t) dM_\star$ and a sample size of $N_k$. The contribution to the covariance matrix from these main branch galaxies is therefore:

$$C_{ij} = \begin{cases} p_{ki}(1 - p_{ki}) N_k w_k^2 & \text{if } i = j \\ -p_{ki} p_{kj} N_k w_k^2 & \text{otherwise,} \end{cases}$$

(10.33)

where $w_k$ is the weight to be assigned to each tree. To compute this covariance requires knowledge of the probabilities, $p_{ki}$. We estimate these directly from the model. To do this, we bin trees into narrow bins of root mass and assume that $p_{ki}$ does not vary significantly across the mass range of each bin. Using all realizations of trees that fall within a given bin, $k$, we can directly estimate $p_{ki}$.

In addition to the main branch galaxies, each tree will contain a number of other galaxies (these will be “satellite” galaxies at $z = 0$, but at higher redshifts may still be central galaxies in their own halos). Tests have established that the number of satellites in halos is well described by a Poisson process. Note that each galaxy contributes Gaussian distribution to the mass function due to modelling of random errors in stellar mass determinations. For main branch galaxies this is simply accounted for when accumulating the probabilities, $p_{ki}$. For satellite galaxies, off-diagonal contributions to the covariance matrix arise as a result, $C_{ij} = w_k f_i f_j$, where $f_i$ is the fraction of the galaxy contributing to bin $i$ of the mass function.

The model likelihood is then computed using:

$$\mathcal{L} = \frac{1}{\sqrt{(2\pi)^n |C|}} \exp \left[ -\frac{1}{2} \Delta C^{-1} \Delta \right],$$

(10.34)

where $\Delta_i = \Phi_{\text{model, } i} - \Phi_{\text{observed, } i}$ is the difference between the model and observed mass functions, and $n$ is the number of points in the mass function histogram.

Computing the large-scale structure contribution to the covariance function requires integration of the non-linear matter power spectrum over the Fourier transform of the survey window function. We use the method of Peacock and Dodds [1996] to determine the non-linear matter power spectrum, because of its simplicity and speed. We have checked that using a more accurate non-linear matter power spectrum (e.g. Lawrence et al. 2010) makes negligible difference to our results.

To find a suitable HOD to describe the galaxies in the Martin et al. [2010] sample we adopt the model of Behroozi et al. [2010]. This is an 11 parameter model which describes separately the numbers of satellite and central galaxies occupying a halo of given mass—the reader is referred to Behroozi et al. [2010] for a complete description of the functional form of this parametric HOD.

To reproduce the mass function of Martin et al. [2010] using this HOD we use the BIE [Weinberg, 2012] to constrain the HOD parameters. We use a likelihood

$$\ln \mathcal{L} = -\frac{1}{2} \Delta \cdot C^{-1} \cdot \Delta^T - \frac{N}{2} \ln(2\pi) - \frac{\ln |C|}{2},$$

(10.35)
where \( N \) is the number of bins in the mass function, \( \mathcal{C} \) is the covariance matrix of the observed mass function, and \( \Delta_i = \phi_i^{\text{HOD}} - \phi_i^{\text{observed}} \). Of course, it is precisely this covariance matrix, \( \mathcal{C} \), that we are trying to compute. We therefore adopt an iterative approach as follows:

1. make an initial estimate of the covariance matrix, assuming that only Poisson errors contribute (the covariance matrix is therefore diagonal, and the terms are easily computed from the measured mass function and the survey volume as a function of stellar mass);

2. find the maximum likelihood parameters of the HOD given the observed mass function and the current estimate of the covariance matrix;

3. using this HOD and the framework of Smith [2012], compute a new estimate of the covariance matrix, including all three contributions;

4. repeat steps 2 and 3 until convergence in the covariance matrix is achieved.

In practice we find that this procedure leads to an HOD and covariance matrix which oscillate between two states in successive iterations. The differences in the covariance matrix are relatively small however, so we choose to conservatively adopt the covariance matrix with the larger values. In future, adding additional constraints to the HOD (as described below) should help mitigate this problem.

Shen et al. (2003) Late-Type Galaxy Size Distribution

This constraint utilizes the distribution of Petrosian half-light radii for \( z \approx 0.07 \) late-type galaxies measured by Shen et al. [2003] from the SDSS. The size function reported by Shen et al. [2003] is converted to the appropriate cosmology for the given GALACTICUS model (assuming that sizes scale as the angular diameter distance, and masses as the square of the luminosity distance).

Given a GALACTICUS model, total stellar masses of model galaxies are adjusted using:

\[
M_\star \rightarrow G S M_\star
\]

where the \( S \) operator is a multiplicative factor accounting for systematic errors in stellar mass determination and is equal to [Behroozi et al., 2010]

\[
\log_{10} S = \mu + \kappa \log_{10} \left( \frac{M_\star}{10^{11.3} M_\odot} \right)
\]

(10.37)

where \( \mu = [\text{diskGalaxySizesSDSSZ0.07MassSystematic0}] \), \( \kappa = [\text{diskGalaxySizesSDSSZ0.07MassSystematic1}] \), and the \( G \) operator is a multiplicative factor drawn from a log-normal distribution of width 0.0806 dex for each galaxy to mimic the effects of random errors on stellar masses (motivated by the statement from Shen et al. [2003] who quote the 95% confidence interval on masses as being \( \pm 40\% \)).

Note: This analysis currently assumes that model galaxies have disk Petrosian half-mass radii of

\[
R_{50} = 1.6676 \frac{1}{\sqrt{2}} \lambda R_{\text{vir}}.
\]

(10.38)

Disk sizes of model galaxies are then adjusted using:

\[
R_{50} \rightarrow G S R_{50}
\]

(10.39)

where the \( S \) operator is a multiplicative factor accounting for systematic errors in radius determination and in determination of radius from halo virial radius and spin and is equal to

\[
\log_{10} S = \mu + \kappa \log_{10} \left( \frac{R_{50}}{1 \text{kpc}} \right)
\]

(10.40)
where $\mu = [\text{iskGalaxySizesSDSSZ0.07RadiusSystematic0}], \kappa = [\text{iskGalaxySizesSDSSZ0.07RadiusSystematic1}]$, and the $G$ operator is a multiplicative factor drawn from a log-normal distribution of width 0.0128 dex for each galaxy to mimic the effects of random errors on disk radii (estimated from the fractional errors reported in the SDSS database).

The model sizes and masses are then used to construct a mass-dependent radius function by binning into a 2-D histogram using the size and mass bins reported by Shen et al. [2003] (modified as described above) as the centers of the bins (with bin boundaries placed at the geometric means of consecutive bin centers).

If the `modelDiscrepancies` option is given, then any multiplicative or additive discrepancies found are applied to the model mass function, and any additional covariance is added to the covariance matrix.

The covariance matrix is computed as

$$C = C_{\text{obs}} + C_{\text{model, random}} + \sum_i C_{\text{discrepancy},i},$$

where $C_{\text{obs}}$ is the covariance matrix of the observational data, $C_{\text{model, random}}$ is the covariance matrix of the model arising from random noise (due to the finite number of trees simulated, and $C_{\text{discrepancy},i}$ is the covariance due to the $i^{th}$ model discrepancy.

The model covariance matrix is estimated using the sample methods as described in §10.5.3. The only difference is that in this case we have a 2-D histogram. This 2-D histogram is “flattened” into a 1-D vector for purposes of likelihood computation however, so covariance matrix estimation proceeds unchanged. (Note that correlations between mass bins are accounted for, in additional to correlations between radius bins.) Since the radius functions of Shen et al. [2003] are normalized to unity at each mass, we must account for this in the covariance matrix. The radius function transforms as:

$$f_{ik} \rightarrow \frac{f_{ik}}{\Delta \log_{10} R \sum_i f_{ik}},$$

where $i$ indexes radius bins, $k$ indexes mass bins, and $\Delta \log_{10} R$ is the width of the radius bin. The Jacobian of this transformation is simply

$$J_{ij} = \frac{\delta_{ij} - f_i}{\Delta \log_{10} R \sum_i f_{ik}}.$$

Therefore, the covariance matrix is modified according to $C \rightarrow JCJ^T$. The same transformation is applied to the covariance matrix of the observed data (for which the reported errors are simply the Poisson errors on each bin).

The model likelihood is then computed using:

$$L = \frac{1}{\sqrt{(2\pi)^n|C|}} \exp \left[ -\frac{1}{2} \Delta C^{-1} \Delta \right],$$

where $\Delta_i = \Phi_{\text{model},i} - \Phi_{\text{observed},i}$ is the difference between the model and observed mass functions, and $n$ is the number of points in the mass function histogram.

### 10.6. Constraint Compilations

To specify which constraints will be applied to a particular model, a compilation file is used. These must be stored in `constraints/compilations`. An example of such a file follows:

```
<constraintCompilation>
```
10.7. Constraint File

Each constraint element specifies one constraint that will be included in this compilation, and must contain a definition element, giving the path of the configuration file for this constraint, and a weight element which allows the relative weight given to each constraint to be varied\(^7\).

10.7. Constraint File

\textsc{Galacticus}'s constraints infrastructure is currently designed to work with a modified version of the \textsc{BIE} to analyze the posterior probability distribution of the model given some compilation of constraints. Note that the standard version of \textsc{BIE} \textit{will not work}\(^8\). To obtain the modified version please contact us. Additionally, the infrastructure currently assumes that you will be performing the analysis on a PBS queue system.

To perform a constraint calculation simply execute the launch script:

```
constraints/bieGalacticus.pl <constraintFile>
```

where constraintFile is a file which describes the details of the calculation to be performed. An example of such a script follows:

```
<constraint>
  <name>verySimplisticToStellarMassFunction</name>
  <workDirectory>my/work/path</workDirectory>
  <scratchDirectory>/scratch/my/scratch/path</scratchDirectory>
  <report>no</report>
  <cpulimit>1200</cpulimit>
  <memoryLimit>32gb</memoryLimit>
  <environment>LD_LIBRARY_PATH=/my/lib:$LD_LIBRARY_PATH</environment>
  <environment>PATH=/my/path:$PATH</environment>
  <baseParameters>parameters.xml</baseParameters>
  <randomize>no</randomize>

  <!-- Compilation of constraints to use -->
  <compilation>stellarMassFunction_SDSS_z0.07.xml</compilation>
  <storeResults>no</storeResults>

  <!-- MCMC method -->
  <threadsPerNode>12</threadsPerNode>
  <threads>300</threads>
```

\(^7\)Note that, if your constraints are computing correct likelihoods, re-weighting them may not be a good idea. \textit{Caveat constrainor}.

\(^8\)The modified version adds new likelihood methods that launch and analyze \textsc{Galacticus} models.
10. Constraining GALACTICUS

<galacticusThreads>12</galacticusThreads>
<galacticusSaveState>no</galacticusSaveState>
<method>differentialEvolution</method>

<!-- MCMC definition -->
<monteCarlo>
  <steps>1000000</steps>
  <stepsSkip>10</stepsSkip>
  <goodSteps>1000</goodSteps>
  <temperingFrequency>100</temperingFrequency>
  <maxTemperature>64.0</maxTemperature>
  <proposalWidth>0.01</proposalWidth>
  <epsWidth>0.001</epsWidth>
  <gamma>0.085875</gamma>
</monteCarlo>

<!-- Use checkpointing -->
<checkpoint>
  <interval>10</interval>
  <autoRestart>false</autoRestart>
</checkpoint>

<!-- Define parameters to vary -->
<parameters>

  <!-- Disk SNe feedback -->
  <parameter>
    <name>diskOutflowTimescaleMinimum</name>
    <prior>
      <distribution>uniform</distribution>
      <mapping>logarithmic</mapping>
      <lowerLimit>1.0e-3</lowerLimit>
      <upperLimit>1.0e-2</upperLimit>
    </prior>
  </parameter>

  <parameter>
    <name>diskOutflowFraction</name>
    <prior>
      <distribution>uniform</distribution>
      <mapping>logarithmic</mapping>
      <lowerLimit>1.0e-1</lowerLimit>
      <upperLimit>1.0e+1</upperLimit>
    </prior>
  </parameter>

</parameters>

<!-- Include the standard WMAP-9 cosmological parameters and priors -->
<x:include href="../../constraints/parameters/wmap9Cosmology.xml"
The following subsections describe each entry in this file.

10.7.1. Computing Environment

- **name**  A name to use for this calculation. It will be used as the name for jobs submitted to the PBS queue for example.

- **workDirectory**  The full path to a directory in which the results (e.g. Markov Chain Monte Carlo (MCMC) chains) will be stored.

- **scratchDirectory** *(Optional)*  The full path to a scratch directory where GALACTICUS model outputs and other temporary data will be written.

- **report**  If set to `yes`, reports additional debugging information during the run.

- **cpulimit**  A CPU time limit for each model evaluation. This can be useful if certain obscure regions of the surveyed parameter space result in unacceptably long run times. Models will be killed after this time and a very low likelihood returned.

- **memoryLimit**  A memory limit to be passed to PBS.

- **environment**  One of more such element can appear. Each specifies the value of an environment variable to be set prior to launching the BIE.

- **storeResults**  If `yes` then the full results (e.g. the quantities computed to compare with observational data) from each model evaluation will be stored. Otherwise, they are discarded after each evaluation. Use `yes` with caution—a typical run might include millions of model evaluations which can quickly lead to huge amounts of data being written to disk.

- **threadsPerNode**  The number of threads to use per node. Typically this would be set to the number of available cores on each node.

- **threads**  The total number of threads (i.e. parallel chains) to use.

- **galacticusThreads**  The number of parallel OpenMP threads to use for each GALACTICUS model. It is recommended that this be set to the number of available cores on each node, and semaphoring (see §sec:Semaphores) be used. In this way, each copy of GALACTICUS on a node will share resources, but as one instance finishes, the others will be able to make use of the freed resources.

- **galacticusSaveState**  If `yes` then GALACTICUS will save its internal state prior to beginning evolution of each merger tree. This is intended for debugging purposes and so should normally be set to `no`.

- **checkpoint**  If present, BIE will be instructed to checkpoint its calculations. The interval (in number of steps) between checkpoints is specified by the `interval` sub-element. If the `autoRestart` sub-element is set to `true` then a second PBS script will be submitted which will restart the calculation once the original job finishes. This second script will submit a further script to take over once it
finishes and so on. This can be useful if your PBS queue has a time limit and you want to keep restarting the calculation until it is done. Use with caution however, some systems forbid automated restarts.

10.7.2. Model Specification

baseParameters Specifies the path to a GALACTICUS parameter file which will be used as the base set of parameter on top of which any parameter variations will be applied.

randomize If yes then each model evaluation will be performed with a different random number seed. Otherwise, the same seed is used in all cases. Experiment shows that changing the random number seed between evaluations can seriously limit the ability of MCMC algorithms to converge.

10.7.3. Constraints

compilation Specifies the path to a compilation file to be used for this analysis.

10.7.4. MCMC Method

method Specifies the MCMC algorithm to be used. Currently accepted values are differentialEvolution and metropolisHastings. See the BIE documentation for full details of these two algorithms.

monteCarlo Specifies the behavior of the MCMC algorithm:
  steps The maximum number of steps to take.
  stepsSkip The number of initial steps to skip before beginning to measure convergence.
  goodSteps The number of steps to perform after convergence is attained.
  temperingFrequency The number of steps between successive temperings (in which the likelihood function is “heated up” to help get chains out of local minima).
  maxTemperature The maximum temperature to use when tempering.
  minTemperatureStates The minimum number of temperatures states when tempering.
  proposalWidth Used for the metropolisHastings algorithm, specifies the width of proposals.
  gamma Used for the differentialEvolution algorithm, specifies the fraction of the vector connecting to chain state to be used as the proposal for another chain.
  epsWidth Used for the differentialEvolution algorithm, specifies the width of the Cauchy distribution added to each proposal to ensure that the chains are positively recurrent.
  grAlpha Set Gelman-Rubin convergence outlier detection confidence, $1 - \alpha$.
  grNOutlier Set number of iterations before applying an outlier test in the Gelman-Rubin convergence statistic.
  grMaxOut Set the maximum number of outliers allowed in the Gelman-Rubin convergence statistic.
  grRhatMax The correlation coefficient threshold for the Gelman-Rubin convergence statistic.

10.7.5. Post-prior

If present, the postPrior block allows the use of the statelog from a previous calculation as the prior on a new calculation (in addition to any explicitly defined priors).

stateLog The full path to the statelog file containing the results of the prior calculation.
10.7. Constraint File
restart If “yes” then the posterior of the previous calculation is used only to sample the initial states of
the new chains. It is not used as a prior on the new calculation. If “no” then the previous posterior
is used as the new prior.
burn The number of steps from the statelog to burn.

10.7.6. Parameters and Priors
The parameters section contains a list of all parameters to be varied in the analysis. Each parameter is
described by one parameter element. That element must contain a name element, which gives the name
of the parameter, and a prior element
distribution The type of distribution to be used for the prior. Currently accepted values are uniform and
normal.
mapping The mapping of the parameter to which the prior should be applied. Currently allowed values are
linear (the prior is on the parameter itself), and logarithmic (the prior is on the logarithm of
the parameter).
lowerLimit The lower limit for the prior.
upperLimit The upper limit for the prior.
mean For normal priors, the mean of the normal distribution.
variance For normal priors, the variance (not the standard deviation) of the normal distribution.
Loading External Parameters/Priors
It is also possible to load parameters and their priors from external files. This is useful to add common
sets of parameters, such as cosmological parameter. To do so, add an element of the form:
<xi:include href="../../constraints/parameters/wmap9Cosmology.xml"
xmlns:xi="http://www.w3.org/2001/XInclude" />
after the parameters section of the constraint file. The href attribute must give the path (relative to the
constraint file, or absolute) to the external parameter file. This file should contain its own parameters
block, describing all parameters to be varied along with their priors.
Derived Parameter Values
It is possible to define parameters in terms of other parameters. Common uses for this include:
• Setting ΩΛ from the value of ΩM to enforce a flat Universe;
• Setting the values of parameters with correlated priors as linear combinations of dummy parameters
for which the priors are independent.
To define a parameter in this way include a parameter element of the form:
<parameter>
<name>sigma_8</name>
<define>0.8178+%cosmology0*0.003817+%cosmology1*0.007931+%cosmology2*0.01002
+%cosmology3*0.001584+%cosmology4*0.002931+%cosmology5*0.001727</define>
</parameter>
Here the define element gives an equation for the parameter in terms of other parameters. All standard mathematical operators and functions (as recognized by Perl) can be used, and other parameters
referenced by using their name prefixed with a “%”.

201


Including External Parameters

Predefined sets of parameters (along with their priors) can be included using the `xi:include` element. For example,

```xml
<xi:include href="../../constraints/parameters/wmap7Cosmology.xml"
  xmlns:xi="http://www.w3.org/2001/XInclude" />
```

will include a set of parameters from the file `../../constraints/parameters/wmap7Cosmology.xml` which defines priors on cosmological parameters consistent with the covariance matrix of the WMAP-7 cosmological constraints [Komatsu et al., 2010].
Part III.

Physical Implementation
In this Part we describe the physical implementation of galaxy formation in GALACTICUS, including all components and their properties and additional output quantities from the code.
11. Definitions and Conventions Used in
GALACTICUS

GALACTICUS adopts various definitions and conventions internally. These are explained below.

11.1. Luminosity Units

Galaxy luminosities are output in the AB magnitude system, such that a luminosity of 1 corresponds to an object of $0^h$ absolute magnitude in the AB magnitude system. This implies that the luminosities are in units of $4.4659 \times 10^{13} \text{ W/Hz}$. 

11.2. Peculiar Velocities

Velocities in GALACTICUS are always physical velocities. When reading merger tree properties (including velocities) from file it is often convenient to store velocities without the Hubble flow contribution, as “peculiar velocities”, in the file—see §A.6 for how to specify whether or not the velocities included in the file include the Hubble flow or not.

If peculiar velocities are stored it is important to use the same definition of peculiar velocity as is used by GALACTICUS. Defining $t$ to be physical time and $x$ to be comoving position, GALACTICUS uses the conventional definition of peculiar velocity in a cosmological context, namely that it is the deviation of the physical velocity from the Hubble flow. Physical coordinates are given by $r = ax$, so the peculiar velocity is

$$v_{pec} = \frac{dr}{dt} - Hr = a \frac{dx}{dt} = \frac{dx}{d\eta}, \quad (11.1)$$

where $d\eta = dt/\alpha$ is conformal time.
12. Node Components

In addition to the implementations described here, each component class has a “null” implementation. Selecting this implementation—which has no properties and does not respond to any events—effectively switches off the relevant component class. Of course, this is safe only if none of the other active implementations expect to get or set properties of the component class (or if they rely on a sensible implementation of that class).

12.1. (Supermassive) Black Hole

12.1.1. “Standard” Implementation

Properties

The standard black hole implementation defines the following properties:

- **mass** The mass of the black hole: \( M_\bullet \) [blackHoleMass].
- **spin** The spin of the black hole, \( j_\bullet \) [blackHoleSpin].
- **radialPosition** The radial position of the black hole: \( r_\bullet \) [blackHoleRadialPosition].

Initialization

Black holes are not initialized, they are created (with a seed mass given by blackHoleSeedMass and zero spin) as needed.

Differential Evolution

In the standard black implementation the mass and spin evolve as:

\[
\begin{align*}
\dot{M}_\bullet &= (1 - \epsilon_{\text{radiation}} - \epsilon_{\text{jet}}) \dot{M}_0 \quad \text{(12.1)} \\
\dot{j}_\bullet &= \dot{j}(M_\bullet, j_\bullet, \dot{M}_0) \quad \text{(12.2)}
\end{align*}
\]

where \( \dot{M}_0 \) is the rest mass accretion rate, \( \epsilon_{\text{radiation}} \) is the radiative efficiency of the accretion flow feeding the black hole, \( \epsilon_{\text{jet}} \) is the efficiency with which accretion power is converted to jet power and \( \dot{j}(M_\bullet, j_\bullet, \dot{M}_0) \) is the spin-up function of that accretion flow (see §17.4.1). The rest mass accretion rate is computed assuming Bondi-Hoyle-Lyttleton accretion from the spheroid gas reservoir (with an assumed temperature of [bondiHoyleAccretionTemperatureSpheroid]) enhanced by a factor of [bondiHoyleAccretionEnhancementSpheroid] and from the host halo (with whatever temperature that hot halo temperature profile specifies; see §13.21) enhanced by a factor of [bondiHoyleAccretionEnhancementHotHalo]. For accretion from the hot halo, the Bondi radius is limited to the outer radius of the hot halo. Additionally, the accretion rate is limited to:

\[
\dot{M}_0, \text{ hot halo, maximum} = \frac{M_{\text{hot}}}{\tau_{\text{sound crossing}}}, \quad \text{(12.3)}
\]

where \( \tau_{\text{sound crossing}} = r_{\text{hot halo outer}}/c_s \) where \( r_{\text{hot halo outer}} \) is the outer radius of the hot halo and \( c_s \) is the speed of sound in the hot halo.
If \[\text{bondiHoyleAccretionHotModeOnly}=\text{true}\] then the accretion occurs only from that fraction of the hot halo gas which was accreted in the “hot mode”, otherwise accretion occurs from the entirety of the hot halo reservoir. In the first case a simple estimate of the hot mode fraction is made:

\[
f_{\text{hot}} = \begin{cases} 
1 & \text{if } x < 0.9 \\
y(x)^2[2y(x) - 3] + 1 & \text{if } 0.9 \leq x \leq 1.0 \\
0 & \text{if } x > 1.0, 
\end{cases} 
\] (12.4)

where \(x = r_{\text{cool}}/r_{\text{virial}}\) and \(y(x) = |x - 0.9|/|1.0 - 0.9|\).

The rest mass accretion rate is removed (as a mass sink) from the spheroid and hot halo components appropriately. The black hole is assumed to cause feedback in two ways:

**Radio-mode** If \[\text{blackHoleHeatsHotHalo}=\text{true}\] then any jet power from the black hole-accretion disk system (see §13.3) is included in the hot halo heating rate providing that the halo is in the slow cooling regime\(^1\) (i.e. if the cooling radius is smaller than the virial radius; see, for example, Benson and Bower 2010);

**Quasar-mode** A mechanical wind luminosity of [Ostriker et al., 2010]

\[
L_{\text{wind}} = \epsilon_{\text{wind}}H(\epsilon_{\text{radiation}}, 1, s)M_0 c^2, 
\] (12.5)

where \(\epsilon_{\text{wind}}=\text{blackHoleWindEfficiency}\) is the black hole wind efficiency, \(s=\text{blackHoleWindEfficiencyScalesWithRadiativeEfficiency}\), and

\[
H(a, b, c) = \begin{cases} 
a & \text{if } c = \text{true} \\
b & \text{if } c = \text{false}, 
\end{cases} 
\] (12.6)

is added to the gas component of the spheroid (which, presumably, will respond with an outflow for example—see §12.5 for details of how specific implementations of the spheroid component respond to the addition of energy) if and only if the wind pressure (at the spheroid characteristic radius) is less than the typical thermal pressure in the spheroid gas [Ciotti et al., 2009], i.e.

\[
\frac{1}{2} \rho_{\text{wind}} V_{\text{wind}}^2 < \frac{3k_B T_{\text{ISM}} \langle \rho_{\text{ISM}} \rangle}{2m_H}. 
\] (12.7)

Since \(\Omega^2 \rho_{\text{wind}} V_{\text{wind}}^2 = L_{\text{wind}}\) where \(\Omega\) is the solid angle of the wind flow, this can be rearranged to give \(\langle \rho_{\text{ISM}} \rangle > \rho_{\text{wind,critical}}\) where

\[
\rho_{\text{wind,critical}} = \frac{2m_H L_{\text{wind}}}{3\Omega^2 V_{\text{wind}} k_B T_{\text{ISM}}}. 
\] (12.8)

This critical wind density is computed at the characteristic radius of the spheroid, \(r_{\text{spheroid}}\), assuming \(V_{\text{wind}} = 10^4\text{km/s}, T_{\text{ISM}} = 10^4\text{K}\) and \(\Omega = \pi\), and the interstellar medium (ISM) density is approximated by

\[
\langle \rho_{\text{ISM}} \rangle = \frac{3M_{\text{gas,spheroid}} r_{\text{spheroid}}^3}{4\pi}. 
\] (12.9)

For numerical ease, the fraction, \(f_{\text{wind}}\), of the wind luminosity added to the spheroid is adjusted smoothly through the \(\rho_{\text{ISM}} \approx \rho_{\text{wind,critical}}\) region according to

\[
f_{\text{wind}} = \begin{cases} 
0 & \text{if } x < 0, \\
3x^2 - 2x^3 & \text{if } 0 \leq x \leq 1, \\
1 & \text{if } x > 1, 
\end{cases} 
\] (12.10)

\(^1\)Specifically, the jet power multiplied by \(f_{\text{hot}}(M_{\text{hot}}/M_{\text{total}})(\Omega_{\text{M}}/\Omega_{\text{h}})^2\) is added to the hot halo heating rate. The dependence on the gas fraction in the hot halo ensures that the heating rate goes smoothly to zero as the hot halo becomes depleted of gas.
where \( x = \rho_{SM}/\rho_{wind,critical} - 1/2 \).

The radial position, \( r_\bullet \), evolves according to the selected radial migration method (see §17.4.1).

Interactions between black hole triplets are accounted for if \( \text{[tripleBlackHoleInteraction]} = \text{true} \) (and if at least three black holes exist within the node of course). In this case the triple is treated as consisting of an inner binary (assumed to be the central black hole and the black hole closest to it) and a third, singleton black hole. When the tertiary black hole reaches a separation of

\[
a_h = \frac{G(M_{\bullet,1} + M_{\bullet,2})}{4\sigma^2} \tag{12.11}
\]

it is assumed to undergo a triple interaction with the binary. Once a triple interaction occurs, no further triple interaction for the specific tertiary black hole can occur unless the host galaxy merges with another galaxy, at which point the black holes from the merging galaxy are eligible for another triple interaction in their new host.

The logic of what happens in a triple black hole interaction is taken from Volonteri et al. [2003]. Labelling the central black hole as 1, its binary partner as 2 and the tertiary black hole as 3, and defining

\[
q_3 = \frac{M_{\bullet,3}}{M_{\bullet,1} + M_{\bullet,2}} \tag{12.12}
\]

then if \( q_3 \leq 2 \) then, if \( M_{\bullet,3} \leq M_{\bullet,2} \) we set

\[
a_3 = \frac{a_2}{1 + 0.4q_3} \tag{12.13}
\]

and define

\[
E_{\text{bind}} = \frac{GM_{\bullet,3}M_{\bullet,1}}{a_3} \tag{12.14}
\]

and

\[
\Delta K = 0.4q_3E_{\text{bind}} \tag{12.15}
\]

\( i = 3 \) and \( j = 2 \).

Otherwise if \( q_3 \leq 2 \) and \( M_{\bullet,3} > M_{\bullet,2} \) we set

\[
a_3 = \frac{a_3}{1 + 0.4q_3} \tag{12.16}
\]

and define

\[
E_{\text{bind}} = \frac{GM_{\bullet,2}M_{\bullet,1}}{a_2} \tag{12.17}
\]

and

\[
\Delta K = 0.4q_3E_{\text{bind}} \tag{12.18}
\]

\( i = 2 \) and \( j = 3 \).

Finally, if \( q_3 > 2 \), then we set

\[
a_3 = 0.53a_3 \tag{12.19}
\]

and define

\[
E_{\text{bind}} = \frac{GM_{\bullet,2}M_{\bullet,1}}{a_2} \tag{12.20}
\]

and

\[
\Delta K = 0.9q_3E_{\text{bind}} \tag{12.21}
\]

\( i = 2, j = 3 \).
Black hole $i$ is identified as the “ejected” hole, with black hole $j$ becoming the new binary member. Therefore

$$M_{\text{\textbullet ejected}} = M_{\text{\textbullet i}}.$$  \hfill (12.22)

and

$$M_{\text{binary}} = M_{\text{\textbullet j}} + M_{\text{\textbullet 1}}.$$  \hfill (12.23)

The imparted velocities of these two systems are

$$V_{\text{\textbullet ejected}} = \left[\frac{2\Delta K}{(1 + M_{\text{\textbullet ejected}}/M_{\text{binary}})M_{\text{\textbullet ejected}}}\right]^{1/2}$$  \hfill (12.24)

and

$$V_{\text{binary}} = \left[\frac{2\Delta K}{(1 + M_{\text{binary}}/M_{\text{\textbullet ejected}})M_{\text{binary}}}\right]^{1/2}.$$  \hfill (12.25)

If

$$\frac{1}{2}V_{\text{\textbullet ejected}}^2 + \Phi(\text{\textbullet ejected}) \geq 0$$  \hfill (12.26)

for either velocity, then that system is ejected from the node. Ejected black holes are removed from the node. If the binary is ejected the central black hole is replaced with a “null”, zero mass placeholder.

**Event Evolution**

*Node mergers:* None.

*Satellite merging:* The black holes in the two merging galaxies can be instantaneously merged, or taken at an initial separation (see §13.47), it is then evolved until reaching zero separation whereupon it is assumed to undergo merger. Properties are computed using the selected black hole binary merger method (see §17.4.1). In addition, the recoil velocity of the new black hole due to gravitational wave emission is computed using the selected method (see §13.49), and if greater than the potential at the center of the galaxy, is assumed to have escaped the galaxy. Black holes which escape the galaxy are simply discarded and no longer tracked. For computational purposes, they are replaced with a “null”, zero mass black hole at the center of the galaxy. If any other black hole comes within a distance

$$a_h = \frac{GM_{\text{\textbullet}}}{4\sigma^2},$$  \hfill (12.27)

where $\sigma$ is approximated to be the virial velocity of the dark matter halo, it is promoted to being the new “central” black hole of the node.

*Node promotion:* None.

**Additional Output**

If the \texttt{[blackHoleOutputAccretion\]} input parameter is set to true, then rest mass accretion rate (in \(M_\odot\) Gyr\(^{-1}\)), jet power (in \(M_\odot\) km\(^2\) s\(^{-1}\) Gyr\(^{-1}\)) and radiative efficiency of the black hole\(^2\) are output as \texttt{blackHoleAccretionRate}, \texttt{blackHoleJetPower} and \texttt{blackHoleRadiativeEfficiency} respectively.

If the \texttt{[blackHoleOutputData\]} input parameter is set to true, then the Masses (in \(M_\odot\)), Spins (for now just a scalar with no direction), final Radius (in Mpc), timescales (in Gyr) until merger, accretion rates (in \(M_\odot\) per Gyr) and radiative Efficiencies of all the black holes in the galaxy are given as outputs.

\(^2\)Technically of the black hole plus accretion disk system.
12.1. (Supermassive) Black Hole

in the `blackHole` section of the output hdf5. This also saves the tree node and merger tree index for further use when using the data.

The outputs of mergers are also automatically saved, as outputs in the `blackHoleMergers` section of the output hdf5. Those outputs are the time at which mergers happened and the mass ratio between the two merging black holes.

12.1.2. “Simple” Implementation

Properties

The simple black hole implementation defines the following property:

mass The mass of the black hole: $M_{\bullet}$.

Initialization

Black holes are not initialized, they are created (with a seed mass given by `blackHoleSeedMass`) as needed.

Differential Evolution

In the simple black hole implementation the mass evolves as:

$$\dot{M}_\bullet = (1 - \epsilon_{\text{wind}}) \epsilon_{BH} \dot{M}_{s,\text{spheroid}}$$

(12.28)

$$\dot{M}_\bullet = \epsilon_{\text{jet}} \dot{M}_{\text{Eddington}}$$

(12.29)

where $\epsilon_{BH}$ is the ratio of rates at which the black hole and stellar spheroid grow. The black hole is assumed to cause feedback in two ways:

**Radio-mode** If $[\text{blackHoleHeatsHotHalo]=true}$ and $[\text{blackHoleAccretesFromHotHalo]=false}$ then a power $\epsilon_{\text{heat}} \epsilon_{BH} \dot{M}_{s,\text{spheroid}} c^2$ where $\epsilon_{\text{heat}} = [\text{blackHoleHeatingEfficiency}]$ is included in the hot halo heating rate providing that the halo is in the slow cooling regime (i.e. if the cooling radius is smaller than the virial radius; see, for example, Benson and Bower 2010) and the accretion rate onto the black hole is reduced by $\epsilon_{\text{heat}} \epsilon_{BH} \dot{M}_{s,\text{spheroid}}$. If $[\text{blackHoleHeatsHotHalo]=true}$ and $[\text{blackHoleAccretesFromHotHalo]=true}$ then a power $\epsilon_{\text{heat}} \dot{M}_{\text{Eddington}} c^2$ is included in the hot halo heating rate providing that the halo is in the slow cooling regime and the accretion rate onto the black hole is increased$^3$ by $\dot{M}_{\text{Eddington}} \epsilon_{\text{heat}} (1 - \epsilon_{\text{jet}})/\epsilon_{\text{jet}}$, where $\epsilon_{\text{jet}} = [\text{blackHoleJetEfficiency}]$;

**Quasar-mode** A mechanical wind luminosity of [Ostriker et al., 2010]

$$L_{\text{wind}} = \epsilon_{\text{wind}} \dot{M}_0 c^2,$$

(12.30)

where $\epsilon_{\text{wind}} = [\text{blackHoleWindEfficiency}]$ is the black hole wind efficiency, is added to the gas component of the spheroid (which, presumably, will respond with an outflow for example).

Event Evolution

**Node mergers:** None.

---

$^3$Note that mass is not removed from the hot halo to compensate, since the accretion rate is independent of the hot halo mass this could lead to negative mass in the halo.
12. Node Components

Satellite merging: The black holes in the two merging galaxies are instantaneously merged. Properties are computed using the selected black hole binary merger method (see §17.4.1).

Node promotion: None.

Additional Output

If the [blackHoleOutputAccretion] input parameter is set to true, then rest mass accretion rate (in $M_\odot/\text{Gyr}$) is output as blackHoleAccretionRate.

12.2. Dynamics Statistics

This class collects statistics related to galaxy dynamics.

12.2.1. “Bars” Implementation

Properties

The “bars” dynamics statistics implementation defines the following properties:

- time: An array of times at which bar statistics are recorded.
- barInstabilityTimescale: The timescale of bar instability at each tabulated time.
- adiabaticRatio: The adiabatic ratio, $a$, of the galaxy at each tabulated time defined as:

$$ a = \left( \frac{r_{\text{peri}}}{v_{\text{peri}}} \right) \left( \frac{2\pi r_{\text{disk}}}{v_{\text{disk}}} \right)^{-1}, \tag{12.31} $$

where $r_{\text{peri}}$ is the pericentric distance of the galaxies orbit, $v_{\text{peri}}$ is the orbital velocity at pericenter, $r_{\text{disk}}$ is the characteristic radius of the disk, and $v_{\text{disk}}$ is the circular velocity at that radius. For non-satellite galaxies the adiabatic ratio is set to $-1$.

Bar statistics are recorded at a frequency given by [dynamicsStatisticsBarsFrequency] times the host halo dynamical time.

Initialization

Component is not initialized, it is created as soon as a disk exists.

Differential Evolution

N/A.

Event Evolution

Node mergers: None.

Satellite merging: None.

Node promotion: None.
12.3. Hot Halo

Additional Output

Bar statistics time series are output for each node to a group Outputs/Output<Ω>/dynamicsStatistics/tree<T> (where <Ω> is the output number and <T> is the tree index) into datasets named time<Ω>, timeScale<Ω>, and adiabaticRatio<Ω> (where <Ω> is the node index).

12.3. Hot Halo

12.3.1. “Very Simple” Implementation

Properties

The very simple hot halo implementation defines the following properties:

- **mass**: The mass of gas in the hot halo: $M_{hot}$ [hotHaloMass].

and the following pipes:

- **Hot_Halo_Cooling_Mass_To**: The net cooling rate of gas mass is sent through this pipe. Any component may claim this pipe and connect to it, allowing it to receive the cooling gas.

- **Hot_Halo_Outflow_Mass_To**: Galactic components that wish to expel gas due to an outflow can send that mass through this pipe, where it will be received into the hot halo component.

Initialization

At initialization, nodes are assigned a mass of gas equal to their own mass, minus the mass of any progenitors, multiplied by $\Omega_b/\Omega_{Matter}$.

Differential Evolution

In the very simple hot halo implementation the hot gas mass and heavy element mass(es) evolves as:

$$\dot{M}_{hot} = -\dot{M}_{cooling} + \dot{M}_{outflow},$$  \hspace{1cm} (12.32)

where $\dot{M}_{cooling}$ is the rate of mass loss from the hot halo due to cooling (see §13.6.2. In the above $\dot{M}_{outflow}$ is the net rate of outflow from any components in the node. For satellite galaxies, the outflow is instead directed to the hot halo of the host node.

Event Evolution

- **Node mergers**: Any hot gas from the merging halo is transferred to its host halo.

- **Satellite merging**: Any hot halo of the satellite node is added to that of the host node and the hot halo component removed from the satellite node.

- **Node promotion**: Any hot halo of the parent node is added to that of the node prior to promotion.

- **Halo formation**: None.
12. Node Components

12.3.2. “Standard” Implementation

Properties

The standard hot halo implementation defines the following properties:

unaccretedMass The mass of gas which could have accreted onto the halo if it always accreted baryons and dark matter in the universal proportion, but which failed to do so (e.g., perhaps due to being photoheated to a high temperature and so being resistant to accretion into shallow potential wells): $M_{\text{failed}}$.

mass The mass of gas in the hot halo: $M_{\text{hot}}$ [hotHaloMass].

angularMomentum The angular momentum of the gas in the hot halo, $J_{\text{hot}}$ [hotHaloAngularMomentum].

abundances The mass(es) of heavy elements in gas in the hot halo, $M_{Z,\text{hot}}$ [hotHalo{abundanceName}].

outflowedMass The mass of gas from outflows in the hot halo: $M_{\text{outflowed}}$ [hotHaloOutflowedMass].

outflowedAngularMomentum The angular momentum of the outflowed gas in the hot halo, $J_{\text{outflowed}}$ [hotHaloOutflowedAngularMomentum].

outflowedAbundances The mass(es) of heavy elements in outflowed gas, $M_{Z,\text{outflowed}}$ [hotHaloOutflowed{abundanceName}].

chemicals The mass(es) of molecules in the hot gas, $M_{\text{chemical}}$ [hotHaloChemicals{chemicalName}].

outerRadius The outer boundary radius of the hot halo: $r_{\text{hot,outer}}$ [hotHaloOuterRadius].

strippedMass The mass of gas which has been stripped from the hot halo (by ram pressure or tidal forces for example): $M_{\text{hot,stripped}}$. This property is computed only if [hotHaloTrackStrippedGas]=true.

strippedAbundances The mass(es) of heavy elements in gas that has been stripped from the hot halo (by ram pressure or tidal forces for example), $M_{Z,\text{hot,stripped}}$. These properties are computed only if [hotHaloTrackStrippedGas]=true.

and the following pipes:

heatSource Energy sent through this pipe is added to the hot halo and used to offset the cooling rate (see below; heat pushed should be in units if $M_{\odot} (\text{km/s})^2 \text{Gyr}^{-1}$).

cooling[Mass|Angular_Momentum|Abundances].To The net cooling rate of gas mass (and metal content and magnitude of angular momentum) is sent through this pipe. Any component may claim this pipe and connect to it, allowing it to receive the cooling gas.

outflowing[Mass|AngularMomentum|Abundances].To Galactic components that wish to expel gas due to an outflow can send that mass (plus metals and angular momentum) through this pipe, where it will be received into the hot halo component.

massSink Removes gas (and proportionate amounts of angular momentum and elements) from the hot gas halo.

Initialization

At initialization, any nodes with no children are assigned a hot halo mass, and failed accreted mass as dictated by the baryonic accretion method (see §13.1) and angular momentum based on the accreted mass and the halo spin parameter.
12.3. Hot Halo

Differential Evolution

In the standard hot halo implementation the hot gas mass and heavy element mass(es) evolves as:

\[
\dot{M}_{\text{failed}} = \dot{M}_{\text{failed accretion}} \\
\dot{M}_{\text{hot}} = \dot{M}_{\text{accretion}} - \dot{M}_{\text{cooling}} + \dot{M}_{\text{outflow, return}} - \dot{M}_{\text{expelled}} - \dot{M}_{\text{hot, stripped}},
\]

\[
\dot{M}_{Z,\text{hot}} = -M_{\text{cooling}} \frac{M_{Z,\text{hot}}}{M_{\text{hot}}} + \dot{M}_{Z,\text{outflow, return}} - \dot{M}_{Z,\text{expelled}} - \dot{M}_{Z,\text{hot, stripped}},
\]

\[
\dot{M}_{\text{chemical}} = -\left[\dot{M}_{\text{cooling}} + \dot{M}_{\text{expelled}} + \dot{M}_{\text{hot, stripped}}\right] \frac{M_{\text{chemical}}}{M_{\text{hot}}} + f_{\text{chemical, outflow}} \dot{M}_{\text{outflow, return}}
\]

\[
\dot{r}_{\text{hot, outer}} = \begin{cases} 
\frac{\rho_{\text{virial}}}{\rho_{\text{hot, outer}}} & \text{if } r_{\text{rp}} < r_{\text{hot, outer}} \\
0 & \text{otherwise.}
\end{cases}
\]

\[
\dot{M}_{\text{hot, stripped}} = -4\pi p_{\text{hot}}(r_{\text{hot, outer}}) r_{\text{hot, outer}}^2 \dot{r}_{\text{hot, outer}} + \dot{M}_{\text{outflows}} f_{\text{outflow, stripped}}
\]

\[
\dot{M}_{Z,\text{hot, stripped}} = -4\pi p_{\text{hot}}(r_{\text{hot, outer}}) r_{\text{hot, outer}}^2 \dot{r}_{\text{hot, outer}}(M_{Z,\text{hot}}/M_{\text{hot}})
\]

\[
+ \dot{M}_{Z,\text{outflows}} f_{\text{outflow, stripped}}
\]

where \( r_{\text{rp}} \) is the ram pressure stripping radius as computed by the \texttt{hotHaloRamPressureStrippingMethod} method (see §13.19), \( \dot{M}_{\text{accretion}} \) is the rate of growth of the hot component due to accretion from the IGM and \( \dot{M}_{\text{failed accretion}} \) is the rate of failed accretion from the IGM (these may include a component due to transfer of mass from the failed to accreted reservoirs) and \( \dot{M}_{\text{cooling}} \) is the rate of mass loss from the hot halo due to cooling (see §13.6.2—cooling rates are computed using the current node if \texttt{hotHaloCoolingFromNode} = \texttt{current node} or from the formation node if that parameter is set to \texttt{formation node} minus any heating rate defined as

\[
\dot{M}_{\text{heating}} = \dot{E}_{\text{input}} / V_{\text{virial}}^2,
\]

where \( \dot{E}_{\text{input}} \) is the rate at which energy is being sent through the “energy input” pipe and \( V_{\text{virial}} \) is the virial velocity of the halo. The net cooling rate is never allowed to drop below zero. If the mass heating rate exceeds the mass cooling rate and \([\texttt{hotHaloExcessHeatDrivesOutflow}] = \texttt{false} \) then the excess energy is not used and \( \dot{M}_{\text{expelled}} = 0 \). Alternatively, if \([\texttt{hotHaloExcessHeatDrivesOutflow}] = \texttt{true} \) then

\[
\dot{M}_{\text{expelled}} = \begin{cases} 
\alpha_{\text{expel}} \dot{M}_{\text{hot}} / \tau_{\text{dynamical}} & \text{if } \dot{M}_{\text{heating}} - \dot{M}_{\text{cool}} > \alpha_{\text{expel}} \dot{M}_{\text{hot}} / \tau_{\text{dynamical}} \\
\dot{M}_{\text{heating}} - \dot{M}_{\text{cool}} & \text{otherwise,}
\end{cases}
\]

where \( \dot{M}_{\text{cool}} \) is the intrinsic cooling rate in the halo (i.e. the cooling rate in the absence of any heating) and \( \alpha_{\text{expel}} = [\texttt{hotHaloExpulsionRateMaximum}] \) limits the maximum rate at which mass can be expelled from the halo.

In the above, \( f_{\text{chemical, return}} \) if the mass fraction of each chemical species in the outflowed gas and is assumed to be equal to that given by the atomic ionization state functions (see §13.24) at the virial temperature and mean density of the halo. Finally, \( \dot{M}_{\text{chemical, reactions}} \) represents the rate of change of masses of chemical species due to chemical and atomic processes and is computed using the chemical rates functions (see §13.32). The angular momentum of the hot gas evolves as:

\[
\dot{J}_{\text{hot}} = \dot{J}_{\text{node}} - M_{\text{cool}} \dot{r}_{\text{cool}} V_{\text{rotate}} + \dot{J}_{\text{outflow, return}} - \dot{M}_{\text{expelled}} J_{\text{hot}} / M_{\text{hot}}.
\]

\[
\text{(12.42)}
\]
where $\dot{M}_{\text{node}}$ and $\dot{J}_{\text{node}}$ are defined in §12.8. For the outflowed components:

\begin{align*}
\dot{M}_{\text{outflowed}} &= -\dot{M}_{\text{outflow,return}} + \dot{M}_{\text{outflows}}(1 - f_{\text{outflow,stripped}}), \quad (12.43) \\
\dot{M}_{Z,\text{outflowed}} &= -\dot{M}_{Z,\text{outflow,return}} + \dot{M}_{Z,\text{outflows}}(1 - f_{\text{outflow,stripped}}) \quad (12.44)
\end{align*}

and:

\[ \dot{J}_{\text{outflowed}} = -\dot{J}_{\text{outflow,return}} + \dot{J}_{\text{outflows}}. \quad (12.46) \]

In the above

\[ \dot{M}|M_{Z}|J_{\text{outflow,return}} = \alpha_{\text{outflow return rate}} \frac{M|M_{Z}|J_{\text{outflows}}}{\tau_{\text{dynamical,halo}}}, \quad (12.47) \]

where $\alpha_{\text{outflow return rate}} = (\text{hotHaloOutflowReturnRate})$ is an input parameter controlling the rate at which gas flows from the outflowed to hot reservoirs, and $\dot{M}|M_{Z}|J_{\text{outflows}}$ are the net rates of outflow from any components in the node.

In the above, $f_{\text{outflow,stripped}}$ is the fraction of outflowing material assumed to be stripped from the halo. The is computed following the algorithm of Font et al. [2008], namely

\[ f_{\text{outflow,stripped}} = \epsilon_{\text{strip}} \frac{M_{\text{hot,outer}}}{M_{\text{hot,virial}}}, \quad (12.48) \]

where $\epsilon_{\text{strip}} = (\text{hotHaloOutflowStrippingEfficiency})$ is an input parameter, $M_{\text{hot,outer}}$ is the mass of hot gas contained within the outer radius of the hot halo and $M_{\text{hot,virial}}$ is the mass of hot gas that would be present if the hot halo extended to the virial radius (i.e. if no stripping had occurred).

A fraction $1 - (\text{hotHaloAngularMomentumLossFraction})$ of the cooling angular momentum rate, $\dot{M}_{\text{cooling}}r_{\text{cool}}V_{\text{rotate}}$, is sent through the Hot_Halo_Cooling_Angular_Momentum pipe.

**Event Evolution**

**Node mergers:** If the starveSatellites parameter is true, then any hot halo properties of the minor node are added to those of the major node and the hot halo component removed from the minor node. Additionally in this case, any material outflowed or stripped from the the satellite galaxy to its hot halo is transferred to the hot halo of the host dark matter halo after each timestep. If stripped mass is being tracked (i.e. if [hotHaloTrackStrippedGas]=true) then any stripped mass is transferred from the satellite galaxy to the hot halo of the host dark matter halo after each timestep. If [hotHaloNodeMergerLimitBaryonFraction]=true then the hot gas content of the merged node is limited such that the total baryon content of the node (including satellites) does not exceed the universal baryon fraction, if possible. Any gas removed to enforce this limit is placed into the unaccreted gas reservoir, from which is may eventually be reaccreted.

**Satellite merging:** If the starveSatellites parameter is false, then any hot halo properties of the satellite node are added to those of the host node and the hot halo component removed from the satellite node.

**Node promotion:** Any hot halo properties of the parent node are added to those of the node prior to promotion.

**Halo formation:** If [hotHaloOutflowReturnOnFormation]=true then all outflowed gas is returned to the hot gas reservoir on halo formation events (see §17.4.3).
12.3.3. “Outflow Tracking” Implementation

Properties
The outflow tracking hot halo implementation extends the “standard” implementation by adding the following properties:

- trackedOutflowMass The mass of gas in the hot halo which arrived there directly via outflow: \( M_{\text{outflow,track}} \) [hotHaloTrackedOutflowMass].
- trackedOutflowAbundances The mass of elements in the hot halo which arrived there directly via outflow: \( M_{Z,\text{outflow,track}} \) [hotHaloTrackedOutflowAbundances].

Initialization
Outflowed masses and element masses are initialized to zero.

Differential Evolution
The tracked outflow masses evolve according to:

\[
\dot{M}_{Z,\text{outflow,track}} = \alpha_{\text{outflow return rate}} \frac{M_{|Z_{\text{outflowed}}}}{\tau_{\text{dynamical,halo}}} - M_{Z,\text{outflow,track}} \dot{M}_{\text{expelled}} / M \tag{12.49}
\]

\[
\dot{M}_{Z,\text{outflow,track}} = \alpha_{\text{outflow return rate}} \frac{M_{|Z_{\text{outflowed}}}}{\tau_{\text{dynamical,halo}}} - M_{Z,\text{outflow,track}} \dot{M}_{\text{expelled}} / M \tag{12.50}
\]

Event Evolution
Node mergers: None.
Satellite merging: None.
Node promotion: None.
Halo formation: None.

12.4. Galactic Disk

12.4.1. “Very Simple” Implementation
This implementation assumes a disk with no structural properties—it consists of just gas and stellar masses.

Properties
The very simple galactic disk implementation defines the following properties:

- massGas The mass of gas in the disk: \( M_{\text{disk.gas}} \) [diskMassGas];
- massStellar The mass of stars in the disk: \( M_{\text{disk.stars}} \) [diskMassStellar].

Initialization
No initialization is performed—disks are created as needed.
12. Node Components

Differential Evolution

In the very simple galactic disk implementation the gas mass evolves as:

\[
\dot{M}_{\text{disk,gas}} = \dot{M}_{\text{cooling}} - \dot{M}_{\text{outflow,disk}} - \dot{M}_{\text{stars,disk}},
\]

where the rate of change of stellar mass is

\[
\dot{M}_{\text{stars,disk}} = \Psi,
\]

with

\[
\Psi = \frac{M_{\text{disk,gas}}}{\tau_{\text{disk,star formation}}}
\]

with \(\tau_{\text{disk,star formation}}\) being the greater of the star formation timescale and \(\Gamma_{\text{disk,star formation,minimum}} \tau_{\text{dyn}}\), where \(\tau_{\text{dyn}}\) is the dynamical time of the halo, and \(\Gamma_{\text{disk,star formation,minimum}} = [\text{diskStarFormationTimescaleMinimum}]\). The outflow rate, \(\dot{M}_{\text{outflow,disk}}\), is computed for the current star formation rate and gas properties by the prescriptions for non-expulsive supernova feedback (see §13.45), but is limited to a maximum of \(\dot{M}_{\text{disk,gas}} / \Gamma_{\text{disk,outflow,minimum}} \tau_{\text{dyn}}\), where \(\Gamma_{\text{disk,outflow,minimum}} = [\text{diskOutflowTimescaleMinimum}]\). This outflow is piped to the hot halo component.

Event Evolution

Node mergers: None

Satellite merging: Disks may be destroyed (or, potentially, created or otherwise modified) as the result of a satellite merging event, as dictated by the selected merger remnant mass movement method (see §13.15.1).

Node promotion: None

12.4.2. “Exponential” Implementation

This implementation assumes a disk with an exponential surface density profile in which stars trace gas.

Properties

The exponential galactic disk implementation defines the following properties:

- **massGas** The mass of gas in the disk: \(M_{\text{disk,gas}}[\text{diskMassGas}]\).
- **abundancesGas** The mass of elements in the gaseous disk: \(M_{Z,\text{disk,gas}}[\text{diskAbundancesGas\{abundanceName\}}]\).
- **massStellar** The mass of stars in the disk: \(M_{\text{disk,stars}}[\text{diskMassStellars}]\).
- **abundancesStellar** The mass of elements in the stellar disk: \(M_{Z,\text{disk,stars}}[\text{diskAbundancesStellar\{abundanceName\}}]\).
- **luminositiesStellar** The luminosities (in multiple bands) of the stellar disk: \(L_{\text{disk,stars}}[\text{diskLuminositiesStellar\{luminosityName\}}]\).
- **angularMomentum** The angular momentum of the disk, \(J_{\text{disk}}[\text{diskAngularMomentum}]\).
- **radius** The radial scale length of the disk, \(R_{\text{disk}}[\text{diskRadius}]\).
- **velocity** The circular velocity of the disk at \(R_{\text{disk}}, V_{\text{disk}}[\text{diskVelocity}]\).
12.4. Galactic Disk

Initialization

No initialization is performed—disks are created as needed.

Differential Evolution

In the exponential galactic disk implementation the gas mass evolves as:

\[ \dot{M}_{\text{disk},\text{gas}} = \dot{M}_{\text{cooling}} - \dot{M}_{\text{outflow, disk}} - \dot{M}_{\text{stars, disk}} - \frac{M_{\text{disk, gas}}}{\tau_{\text{bar}}} - \dot{M}_{\text{ram pressure}} - \frac{M_{\text{disk, gas}}}{M_{\text{disk, gas}} + M_{\text{disk, stars}}} \dot{M}_{\text{tidal}}, \]  

(12.54)

where the rate of change of stellar mass is

\[ \dot{M}_{\text{stars, disk}} = \Psi - \dot{R} - \frac{M_{\text{stars, disk}}}{\tau_{\text{bar}}} - \frac{M_{\text{disk, stars}}}{M_{\text{disk, gas}} + M_{\text{disk, stars}}} \dot{M}_{\text{tidal}}, \]  

(12.55)

with

\[ \Psi = \frac{M_{\text{disk, gas}}}{\tau_{\text{disk, star formation}}} \]  

(12.56)

with \( \tau_{\text{disk, star formation}} \) being the star formation timescale and \( \dot{R} \) is the rate of mass recycling from stars and \( \tau_{\text{bar}} \) is a bar instability timescale (see §13.9). The mass removed from the disk by the bar instability mechanism is added to the active spheroid component. Element abundances (including total metals) evolve according to:

\[ \dot{M}_{Z, \text{disk, gas}} = \dot{M}_{Z, \text{cooling}} - \dot{M}_{Z, \text{outflow, disk}} - \dot{M}_{Z, \text{stars, disk}} + \dot{y} - \frac{M_{Z, \text{disk, gas}}}{M_{\text{disk, gas}}} \dot{M}_{\text{ram pressure}} - \frac{M_{Z, \text{disk, gas}}}{M_{\text{disk, gas}} + M_{\text{disk, stars}}} \dot{M}_{\text{tidal}}, \]  

(12.57)

and

\[ \dot{M}_{Z, \text{stars, disk}} = \Psi \frac{M_{Z, \text{disk, gas}}}{M_{\text{disk, gas}}} - \dot{R}_Z - \frac{M_{Z, \text{disk, stars}}}{M_{\text{disk, gas}} + M_{\text{disk, stars}}} \dot{M}_{\text{tidal}}, \]  

(12.58)

where \( \dot{y} \) is the rate of element yield from stars and \( \dot{R}_Z \) is the rate of element recycling. The angular momentum evolves as:

\[ \dot{J}_{\text{disk}} = \dot{J}_{\text{cooling}} - \left[ \dot{M}_{\text{outflow, disk}} + \frac{M_{\text{disk, gas}} + M_{\text{disk, stars}}}{\tau_{\text{bar}}} + \dot{M}_{\text{ram pressure}} + \dot{M}_{\text{tidal}} \right] \frac{J_{\text{disk}}}{M_{\text{disk, gas}}}, \]  

(12.59)

The outflow rate, \( \dot{M}_{\text{outflow, disk}} \), is computed for the current star formation rate and gas properties by the stellar properties subsystem (see §13.40) and prescriptions for expulsive and non-expulsive supernova feedback (see §13.46 and §13.45 respectively), but is not allowed to exceed \( M_{\text{gas, disk}}/\alpha_{\text{outflow minimum, disk}} \tau_{\text{disk, dynamical}} \), where \( \tau_{\text{disk, dynamical}} = R_{\text{disk}}/V_{\text{disk}} \) is the dynamical time of the disk and \( \alpha_{\text{outflow minimum, disk}} \) is the shortest timescale (in units of the dynamical timescale) on which gas can be removed from the disk. This limit prevents the disk being depleted on arbitrarily short timescales. The non-expulsive component of the outflow is piped to the hot halo component. The ram pressure and tidal mass loss rates, \( \dot{M}_{\text{ram pressure}} \) and \( \dot{M}_{\text{tidal}} \), are computed using the selected methods (see §13.33 and §13.51 respectively).

Event Evolution

Node mergers: None

Satellite merging: Disks may be destroyed (or, potentially, created or otherwise modified) as the result of a satellite merging event, as dictated by the selected merger remnant mass movement method (see §13.15.1).

Node promotion: None
12. Node Components

Additional Output

If the [diskOutputStarFormationRate] input parameter is set to true, then the instantaneous star formation rate in the disk (in units of \(M_\odot \text{ Gyr}^{-1}\)) will be included in the output, as diskStarFormationRate.

Structure

The radial size of the disk is found solving for equilibrium (i.e. the radius is such that the angular momentum of material at that radius is sufficient to provide rotational support) at the specified [diskStructureSolverRadius] which is given in units of the disk scale length. In converting from the mean specific angular momentum of the disk to the angular momentum at that radius, a flat rotation curve is assumed, i.e.:

\[
j(r)/\langle j \rangle = rV/\int_0^\infty 2\pi r'\Sigma(r')r'Vdr'
\]

\[
j(r)/\langle j \rangle = r/2r_{\text{disk}}.
\]

(12.60)

The option [diskRadiusSolverCole2000Method], if set to true, alters this behavior to match that of the structure solver used by Cole et al. [2000], in which adiabatic contraction of the dark matter halo is solved for assuming that the disk has a spherical mass distribution. The specific angular momentum passed to the structure solver will be modified as follows in this case:

\[
j(r) \rightarrow [j^2(r) - (V_{\text{disk}}^2(r)r^2 - GM_{\text{disk}}(<r)r)]^{1/2},
\]

(12.61)

where \(V_{\text{disk}}\) is the rotation curve in the plane of an infinitely thin exponential disk. This adjustment accounts for the difference between a thin disk and spherical mass distribution. Note that in this case (as in Cole et al. 2000) the resulting disk will not precisely satisfy \(j(r) = rV_c(r)\) where \(V_c(r)\) is the net rotation curve.

12.5. Galactic Spheroid

12.5.1. “Standard” Implementation

The standard spheroid implementation assumes a spheroid density profile described by a single length scale in which stars trace gas. Currently, two options for the density profile are allowed:

- hernquist: Assumes a Hernquist profile [Hernquist, 1990] for the spheroidal component of a galaxy.
- sersic: Assumes a Sérsic profile (Sérsic 1963; see also Mazure and Capelato 2002) for the spheroidal component of a galaxy in which stars trace gas. The projected density profile of the spheroid is given by:

\[
\Sigma(R) \propto \exp\left(-b_nR^{1/n}\right),
\]

(12.62)

where the Sérsic index, \(n = [\text{spheroidSersicIndex}]\) and the coefficient \(b_n = 2.303(0.8689n - 0.1447)\) Wadadekar et al. [1999]. The 3D density distribution for a given \(n\) is inferred by solving the relevant inverse Abel integral.

\[\text{4}\text{The spheroid density distribution is handled internally using a massDistribution object (see §17.5.4). As such, any mass distribution implemented as an extension of the massDistribution class (and which is described by a single length scale) could be trivially added to the standard spheroid component.}\]
Properties

The standard galactic spheroid implementation defines the following properties:

- **masGas** The mass of gas in the spheroid: \( M_{_{\text{spheroid,gas}}} \)
- **abundancesGas** The mass of elements in the gaseous spheroid: \( M_{_{\text{Z,spheroid,gas}}} \)
- **massStellar** The mass of stars in the spheroid: \( M_{_{\text{spheroid,stars}}} \)
- **abundancesStellar** The mass of elements in the stellar spheroid: \( M_{_{\text{Z,spheroid,stars}}} \)
- **luminositiesStellar** The luminosities (in multiple bands) of the stellar spheroid: \( L_{_{\text{spheroid,stars}}} \)
- **angularMomentum** The pseudo-angular momentum\(^5\) of the spheroid, \( J_{_{\text{spheroid}}} \)
- **radius** The radial scale length of the spheroid, \( r_{_{\text{spheroid}}} \)
- **velocity** The circular velocity of the spheroid at \( r_{_{\text{spheroid}}} \), \( V_{_{\text{spheroid}}} \)
- **energyInput** Energy sent through this pipe is added to the gas of the spheroid and will result in an outflow (see below). Input energy should be in units of \( M_\odot \text{ km}^2 \text{ s}^{-2} \text{ Gyr}^{-1} \) and must be positive (energy cannot be removed from the gas via this pipe).
- **massGasSink** Removes gas (and proportionate amounts of angular momentum and elements) from the spheroid gas. Removed mass should be in units of \( M_\odot \) and must be positive (a negative mass sink would add mass to the spheroid which is not allowed via this pipe).

Initialization

No initialization is performed—spheroids are created as needed.

---

\(^5\) Effectively the angular momentum that the spheroid would have, were it rotationally supported rather than pressure supported.
Differential Evolution

In the standard galactic spheroid implementation the gas mass evolves as:

\[
\dot{M}_{\text{gas, sph}} = -\dot{M}_{\text{outflow, sph}} - \dot{M}_{\text{stars, sph}} - \dot{M}_{\text{rampressure}} - \frac{\dot{M}_{\text{gas, sph}}}{\dot{M}_{\text{gas}} + \dot{M}_{\text{stars, sph}}} \dot{M}_{\text{tidal}},
\]

where the rate of change of stellar mass is

\[
\dot{M}_{\text{stars, sph}} = \Psi - \dot{R} - \frac{\dot{M}_{\text{stars, sph}}}{\dot{M}_{\text{gas}} + \dot{M}_{\text{stars, sph}}} \dot{M}_{\text{tidal}},
\]

with

\[
\Psi = \frac{\dot{M}_{\text{gas, sph}}}{\tau_{\text{star formation, sph}}},
\]

with \(\tau_{\text{star formation, sph}}\) being the star formation timescale and \(\dot{R}\) is the rate of mass recycling from stars. Element abundances (including total metals) evolve according to:

\[
\dot{M}_{Z,\text{gas, sph}} = -\dot{M}_{Z,\text{outflow, sph}} - \dot{M}_{Z,\text{stars, sph}} + \dot{y} - \frac{\dot{M}_{Z,\text{gas, sph}}}{\dot{M}_{\text{gas}}} \dot{M}_{\text{rampressure}} - \frac{\dot{M}_{Z,\text{gas, sph}}}{\dot{M}_{\text{gas}} + \dot{M}_{\text{stars, sph}}} \dot{M}_{\text{tidal}},
\]

and

\[
\dot{M}_{Z,\text{stars, sph}} = \Psi \frac{\dot{M}_{Z,\text{gas, sph}}}{\dot{M}_{\text{gas}}} - \dot{R} Z - \frac{\dot{M}_{Z,\text{stars, sph}}}{\dot{M}_{\text{gas}} + \dot{M}_{\text{stars, sph}}} \dot{M}_{\text{tidal}},
\]

where \(\dot{y}\) is the rate of element yield from stars and \(\dot{R} Z\) is the rate of element recycling. The angular momentum evolves as:

\[
J_{\text{sph}} = -(\dot{M}_{\text{outflow, sph}} + \dot{M}_{\text{rampressure}} + \dot{M}_{\text{tidal}}) \frac{J_{\text{sph}}}{\dot{M}_{\text{gas}} + \dot{M}_{\text{stars, sph}}} + |T| (\dot{M}_{\text{gas}} + \dot{M}_{\text{stars, sph}}) R_{\text{sph}}^2.
\]

The outflow rate, \(\dot{M}_{\text{outflow, sph}}\), is computed for the current star formation rate and gas properties by the stellar properties subsystem (see §13.40) and prescriptions for expulsive and non-expulsive supernova feedback (see §13.46 and §13.45 respectively), with an additional contribution given by

\[
\dot{M}_{\text{outflow, sph}} = \beta_{\text{energy, sph}} \frac{\dot{E}_{\text{gas, sph}}}{V_{\text{sph}}^2},
\]

where \(\beta_{\text{energy, sph}}\) is an input parameter, and \(\dot{E}_{\text{gas, sph}}\) is any input energy sent through the Tree_Node_Spheroid_Gas_Energy_Input pipe, but is not allowed to exceed \(M_{\text{gas, sph}}/\alpha_{\text{outflow, min, sph}}\), where \(\tau_{\text{dynamical, sph}} = R_{\text{sph}}/V_{\text{sph}}\) is the dynamical time of the spheroid and \(\alpha_{\text{outflow, min, sph}}\) is the shortest timescale (in units of the dynamical timescale) on which gas can be removed from the spheroid. This limit prevents the spheroid being depleted on arbitrarily short timescales. The non-expulsive component of the outflow is piped to the hot halo component. The ram pressure and tidal mass loss rates, \(\dot{M}_{\text{rampressure}}\) and \(\dot{M}_{\text{tidal}}\), are computed using the selected methods (see §13.33 and §13.51 respectively). The final term in 12.69 accounts for tidal heating of the spheroid due to the tidal field, \(T\).

\footnote{There may be an additional contribution to the mass and angular momentum rates of change in the spheroid due to material transferred from the disk component via the bar instability mechanism (see §12.4.2). This is not included here as it is not intrinsic to this specific spheroid implementation—it is handled explicitly by the disk component and so applies equally to any spheroid component implementation.}
12.6. Host History

Event Evolution

Node mergers: None

Satellite merging: Spheroids may be created as the result of a satellite merging event, as dictated by the selected merger remnant mass movement method (see §17.4.1).

Node promotion: None.

Additional Output

If the \([\text{spheroidOutputStarFormationRate}]\) input parameter is set to true, then the instantaneous star formation rate in the spheroid (in units of \(M_\odot \text{Gyr}^{-1}\)) will be included in the output, as \(\text{spheroidStarFormationRate}\).

12.6. Host History

This component class tracks various properties of the host halo of each node over time.

12.6.1. “Standard” Implementation

Properties

The standard host history implementation defines the following properties:

- \(\text{hostMassMaximum}\): The maximum mass of the host halo in which this node has ever been hosted: \(M_{\text{host,maximum}}\).

Initialization

The maximum host mass is initialized to the initial host mass, or to \(-1\) for initially isolated nodes.

Differential Evolution

N/A.

Event Evolution

Post evolution: Immediately after evolution of the node, the maximum host mass is set to the maximum of the current host mass and the previous maximum host mass.

Node mergers: As for “post evolution”.

Satellite merging: None.

Node promotion: As for “post evolution”.

12.7. Mass Flow Statistics

This component class tracks the flow of mass between different components of a node.
12.7.1. “Standard” Implementation

Properties
The standard mass flow statistics implementation defines the following properties:

\textbf{cooledMass} The cumulative mass of gas which has cooled directly onto this galaxy: \( M_{\text{cooled}} \).

Initialization
The cooled mass is initialized to zero.

Differential Evolution
The cooled mass increases at a rate equal to the cooling rate onto the galaxy:
\[
\dot{M}_{\text{cooled}} = \dot{M}_{\text{cool}}.
\] (12.71)

Event Evolution
Post evolution: None.

Node mergers: None.

Satellite merging: None.

Node promotion: None.

12.8. Basic Properties

Basic properties are the total mass of a \textit{node} and the cosmic time at which it currently exists.

12.8.1. “Non-evolving” Implementation

Properties
The non-evolving basic properties implementation defines the following properties:

mass The total mass of the node: \( M_{\text{node}} \).

time The time at which the node is defined: \( t_{\text{node}} \).

timeLastIsolated The time at which the node was last an isolated halo (i.e. not a subhalo): \( t_{\text{node}} \).

Initialization
All basic properties are required to be initialized by the merger tree construction routine.

Differential Evolution
Properties are evolved according to:
\[
\dot{M}_{\text{node}} = 0 \quad \text{ (12.72)}
\]
\[
\dot{t}_{\text{node}} = 1. \quad \text{ (12.73)}
\]
Event Evolution

Node mergers: None.

Satellite merging: None.

Node promotion: $M_{\text{node}}$ is updated to the node mass of the parent prior to promotion.

12.8.2. “Standard Implementation

Properties

The standard basic properties implementation defines the following properties:

- **mass** The total mass of the node: $M_{\text{node}} \ [\text{basicMass}]$.
- **time** The time at which the node is defined: $t_{\text{node}}$.
- **timeLastIsolated** The time at which the node was last an isolated halo (i.e. not a subhalo): $[\text{basicTimeLastIsolated}]$.

Initialization

All basic properties are required to be initialized by the merger tree construction routine.

Differential Evolution

Properties are evolved according to:

\[
\dot{M}_{\text{node}} = \begin{cases} 
\frac{M_{\text{node, parent}} - M_{\text{node}}}{t_{\text{node, parent}} - t_{\text{node}}} & \text{if primary progenitor} \\
0 & \text{otherwise}, 
\end{cases} 
\]  

(12.74)

\[
\dot{t}_{\text{node}} = 1, 
\]  

(12.75)

where the “parent” subscript indicates a property of the parent node in the merger tree.

Event Evolution

Node mergers: None.

Satellite merging: None.

Node promotion: $M_{\text{node}}$ is updated to the node mass of the parent prior to promotion.

12.8.3. “Standard-Tracking Implementation

Properties

The standard-tracking basic properties implementation extends the standard implementation and defines the following additional properties:

- **massMaximum** The maximum total mass of the node achieved prior to the current time along this node’s branch: $M_{\text{node, maximum}} \ [\text{basicMassMaximum}]$. 


12. Node Components

**Initialization**
The maximum mass along each branch is computed from the merger tree structure at initialization time.

**Differential Evolution**
None.

**Event Evolution**

*Node mergers:* None.

*Satellite merging:* None.

*Node promotion:* $M_{\text{node, maximum}}$ is updated to the node maximum mass of the parent prior to promotion.

12.9. Position
The position component implements the position and velocity of each galaxy. See §11.2 for important notes on velocity definitions in **Galacticus**.

12.9.1. “Preset” Implementation

**Properties**
The preset position implementation defines the following properties:

*position* The 3-D position of the node: $x \{\text{positionPosition}[X|Y|Z]\}.$

*velocity* The 3-D velocity of the node: $v \{\text{positionVelocity}[X|Y|Z]\}.$

*positionHistory* The history of the node’s position in 6-D phase space, usually used for satellite nodes.

**Initialization**
None—all properties are assumed to have been preset, usually by the merger tree construction routine.

**Differential Evolution**
None. Positions and velocities do not evolve for a given node. When output, if a 6-D position history is available than the position and velocity from the history entry closest to the output time will be used.

**Event Evolution**

*Node mergers:* None.

*Satellite merging:* None.

*Node promotion:* The position and velocity are updated to those of the parent node.

---

7While interpolation could be used this is usually a bad idea. For nodes that are satellites in a halo for example, no simple interpolation algorithm can correctly account for the complex orbital dynamics by which the position and velocity is actually evolving.
12.10. Satellite Orbit

This component tracks the orbital properties of subhalos.

12.10.1. “Preset” Implementation

Properties

The preset satellite orbit implementation defines the following properties:

mergeTime The time until the satellite will merge with its host: \( t_{\text{satellite,merge}} \) [\( \text{satelliteMergeTime} \)].

timeOfMerging The cosmological time at which the satellite will merge with its host: \( T_{\text{satellite,merge}} \).

boundMass The remaining, total bound mass of the satellite (this property is read only—it is determined from the \( \text{boundMassHistory} \) property).

boundMassHistory A history time-series of the total bound mass of the satellite.

virialOrbit The orbit (a \( \text{keplerOrbit} \) object; see §17.5.1) of the satellite at virial orbit crossing.

Note that the mergeTime and timeOfMerging effectively provide the same information. For that reason, setting one of them will automatically set the other accordingly.

Initialization

None. This method assumes that merging times and bound mass histories will be set externally (usually when the merger tree is constructed).

Differential Evolution

None.

Event Evolution

Node mergers: None.

Satellite merging: None.

Node promotion: None.

12.10.2. “Very Simple” Implementation

Properties

The simple satellite orbit implementation defines the following properties:

mergeTime The time until the satellite will merge with its host: \( t_{\text{satellite,merge}} \) [\( \text{satelliteMergeTime} \)].

Initialization

None.
### Differential Evolution

Properties are evolved according to:

\[
\dot{t}_{\text{satellite,merge}} = -1.
\]  

(12.76)

### Event Evolution

**Node mergers:** The *component* is created and the time to merging is assigned a value.

**Satellite merging:** None.

**Node promotion:** Not applicable (component only exists for satellite nodes).

---

#### 12.10.3. “Simple” Implementation

**Properties**

The simple satellite orbit implementation defines the following properties:

- **mergeTime** The time until the satellite will merge with its host: \(t_{\text{satellite,merge}} = \text{satelliteMergeTime}\).
- **boundMass** The remaining, total bound mass of the satellite: \(M_{\text{node,bound}} = \text{satelliteBoundMass}\).
- **virialOrbit** The orbit (returned as a *keplerOrbit* object; see §17.5.1) of the satellite at the point of virial radius crossing.

**Initialization**

None.

**Differential Evolution**

Properties are evolved according to:

\[
\dot{t}_{\text{satellite,merge}} = -1,
\]  

(12.77)

with \(\dot{M}_{\text{node,bound}}\) set to the rate given by the darkMatterHaloMassLossRateMethod method (see §17.4.1). The virial orbit is a fixed quantity and does not evolve.

**Event Evolution**

**Node mergers:** The *component* is created and the time to merging is assigned a value. The bound mass is set to the current total mass of the node. If satelliteOrbitStoreOrbitalParameters=true then a virial orbit is selected (unless one has already been set for the node) using the virialOrbitsMethod (see §17.4.1) and stored (otherwise, a new virial orbit will be computed—possibly at random—each time the virial orbit is requested). If [satelliteOrbitResetOnHaloFormation]=true then satellite orbits will be reset on halo formation events (see §12.16).

**Satellite merging:** None.

**Node promotion:** Not applicable (component only exists for satellite nodes).
12.10.4. “Orbiting” Implementation

Properties

The orbiting satellite orbit implementation defines the following properties:

position The 3-dimensional position of the satellite relative to its host: \( r = (x, y, z) \).

velocity The 3-dimensional velocity of the satellite relative to its host: \( v = (v_x, v_y, v_z) \).

mergeTime The time until the satellite will merge with its host: \( t_{\text{satellite,merge}} [\text{satelliteMergeTime}] \).

boundMass The remaining, total bound mass of the satellite: \( M_{\text{node,bound}} [\text{satelliteBoundMass}] \).

virialOrbit The orbit (returned as a \textit{keplerOrbit} object; see §17.5.1) of the satellite at the point of virial radius crossing.

tidalTensorPathIntegrated The time integral of the tidal tensor along the orbit of the satellite from initialization: \( G_{ij} \).

tidalHeatingNormalized The tidal heating energy per radius squared desposited into the satellite: \( Q_{\text{tidal}} \).

Initialization

The satellite position, if not already assigned, is selected such that its magnitude is set according to its virial orbit parameters and its direction is chosen at random assuming an isotropic distribution. The satellite velocity is similarly selected. The integrated tidal tensor, \( G_{ij} \), is initialized to the null tensor, while \( Q_{\text{tidal}} \) is initialized to zero.

Differential Evolution

Properties are evolved according to:

\[
\begin{align*}
\dot{r} &= v \quad (12.78) \\
\dot{v} &= -\frac{GM_{\text{host}}(<r)r}{r^3} + a_{DF} \quad (12.79) \\
\dot{G}_{ij} &= g_{ij} - G_{ij}/T_{\text{orb}} \quad (12.80)
\end{align*}
\]

with \( r = |r| \), \( a_{DF} \) set to the rate given by the \textit{satelliteDynamicalFrictionMethod} (see §13.34), \( g_{ij} \) being the tidal tensor, \( T_{\text{orb}} \) being the satellite’s orbital period, \( M_{\text{node,bound}} \) set to the rate given by the \textit{satelliteTidalStrippingMethod} (see §13.36), and \( Q_{\text{tidal}} \) set to the rate given by the \textit{satelliteTidalHeatingMethod} (see §13.37). Note that in the evolution of \( G_{ij} \) a decay term is artificially introduced such that the integration of \( g_{ij} \) is effectively over just the previous orbit (i.e. over the previous tidal shock).

Event Evolution

Node mergers: The component is created and the time to merging is given a value of -1 to indicate that the satellite is not about to merge as unmerged. Once the satellite satisfies one of the two merging conditions:

\[
\begin{align*}
 r &< R_{\text{host}} + R_{\text{satellite}} \quad (12.81) \\
 M_{\text{node,bound}} &< f_d M_{\text{node,basic}} \quad (12.82)
\end{align*}
\]
with $R$ being the node’s half-mass radius, $M_{node,\text{basic}}$ being the node’s initial mass, and $f_d = \text{[satelliteOrbitingDestructionMassFraction]}$, the node is considered merged and the time to merging is set to zero. The bound mass is set to the current total mass of the node. A virial orbit is selected using the \text{virialOrbitsMethod} (see §17.4.1).

\textit{Satellite merging:} None.

\textit{Node promotion:} Not applicable (component only exists for satellite nodes).

### 12.11. N-body Halo Properties

This component is intended for tracking of miscellaneous properties derived from N-body simulation merger trees.

#### 12.11.1. “Standard” Implementation

\textbf{Properties}

The standard N-body component implementation defines the following properties:

- \texttt{particleCount} The number of particles in the N-body halo [\texttt{nbodyParticleCount}].
- \texttt{velocityMaximum} The maximum of the N-body halo’s rotation curve (in units of km/s) [\texttt{nbodyVelocityMaximum}].
- \texttt{velocityDispersion} The velocity dispersion of the N-body halo (in units of km/s) [\texttt{nbodyVelocityDispersion}].

\textbf{Initialization}

The N-body properties of each node are assumed to have been preset prior to merger tree initialization. They are assumed to remain constant along each branch of the tree.

\textbf{Differential Evolution}

None.

\textbf{Event Evolution}

- \textit{Node mergers:} None.
- \textit{Satellite merging:} None.
- \textit{Node promotion:} The properties are updated to equal those of the parent node.

### 12.12. Dark Matter Halo Spin

#### 12.12.1. “Random” Implementation

\textbf{Properties}

The random dark matter halo spin implementation defines the following properties:

- \texttt{spin} The spin parameter of the halo: $\lambda$ [\texttt{spinSpin}].
12.12. Dark Matter Halo Spin

**Initialization**

The spin parameter of each node, if not already assigned, is selected at random from a distribution of spin parameters. This value is assigned to the earliest progenitor of the halo traced along its primary branch. The value is then propagated forward along the primary branch until the node mass exceeds that of the node for which the spin was selected by a factor of \([\text{randomSpinResetMassFactor}]\), at which point a new spin is selected at random, and the process repeated until the end of the branch is reached.

**Differential Evolution**

The spin parameter does not evolve.

**Event Evolution**

*Node mergers:* None.

*Satellite merging:* None.

*Node promotion:* The spin is updated to equal that of the parent node. (The two will differ only if this is a case where the new halo node was sufficiently more massive than the node for which a spin was last selected that a new spin value was chosen.)

12.12.2. “Preset” Implementation

**Properties**

The preset dark matter halo spin implementation defines the following properties:

- **spin**  The spin parameter of the halo: \(\lambda [\text{spinSpin}]\).
- **spinGrowthRate**  The growth rate spin parameter of the halo (in units of Gyr\(^{-1}\)).

**Initialization**

The spin parameter of each node is assumed to have been preset prior to merger tree initialization. The growth rate is computed assuming linear growth with time along each branch.

**Differential Evolution**

The spin parameter evolves linearly with time between node and parent node.

**Event Evolution**

*Node mergers:* None.

*Satellite merging:* None.

*Node promotion:* The spin and growth rate are updated to equal those of the parent node.
12. Node Components

12.12.3. “Preset3D” Implementation

The type extends the preset spin component implementation, and so provides all properties of that implementation plus those described below.

Properties

The preset dark matter halo spin implementation defines the following properties:

- `spinVector` The spin vector of the halo: \( \lambda \) [spinSpinVector].

- `spinVectorGrowthRate` The growth rate of the spin vector of the halo (in units of Gyr\(^{-1}\)).

Initialization

The spin vector of each node is assumed to have been preset prior to merger tree initialization. The growth rate is computed assuming linear growth with time along each branch.

Differential Evolution

The spin parameter evolves linearly with time between node and parent node.

Event Evolution

- **Node mergers:** None.

- **Satellite merging:** None.

- **Node promotion:** The spin vector and growth rate are updated to equal those of the parent node.

12.13. Dark Matter Profile

This component stores dynamic properties associated with dark matter halo density profiles.

12.13.1. “Scale” Implementation

Properties

The scale dark matter profile implementation defines the following properties:

- `scale` The scale length of the density profile [darkMatterProfileScale];

- `scaleGrowthRate` The growth rate of the scale length of the density profile.

Initialization

The scale length of each node, if not already assigned, is assigned using the concentration parameter function (see §13.8.3), but is not allowed to drop below [darkMatterProfileMinimumConcentration], such that the scale length is equal to the virial radius divided by that concentration.

Differential Evolution

The scale radius grows linearly with time to interpolate between the scale radii of child and parent halos.
Event Evolution

Node mergers: None.

Satellite merging: None.

Node promotion: None.

12.13.2. “Scale Preset” Implementation

Properties

The “scale preset” dark matter profile implementation defines the following properties:

- **scale** The scale length of the density profile [darkMatterProfileScale];
- **scaleGrowthRate** The growth rate of the scale length of the density profile.

Initialization

The scale length of each node is assumed to be assigned during tree construction.

Differential Evolution

The scale radius grows linearly with time to interpolate between the scale radii of child and parent halos.

Event Evolution

Node mergers: None.

Satellite merging: None.

Node promotion: None.

12.13.3. “Scale+Shape” Implementation

Properties

The scale+shape dark matter profile implementation defines the following properties:

- **scale** The scale length of the density profile [darkMatterProfileScale];
- **scaleGrowthRate** The growth rate of the scale length of the density profile.
- **shape** A shape parameter describing the density profile [darkMatterProfileShape];
- **shapeGrowthRate** The growth rate of the shape parameter of the density profile.

Initialization

The scale length of each node, if not already assigned, is assigned using the concentration parameter function (see §13.8.3), but is not allowed to drop below [darkMatterProfileMinimumConcentration], such that the scale length is equal to the virial radius divided by that concentration. The shape parameter of each node is assigned using the dark matter profile shape function (see §17.4.1).
12. Node Components

**Differential Evolution**

The scale radius and shape parameter of each node grow linearly with time to interpolate between the scale radii and shape parameters of child and parent halos.

**Event Evolution**

*Node mergers:* None.

*Satellite merging:* None.

*Node promotion:* None.


This component records statistics associated with galaxy merging.


**Properties**

The standard merging statistics implementation defines the following properties:

- `galaxyMajorMergerTime` The time of the last major merger associated with this galaxy, output as the time since the last major merger [mergingStatisticsGalaxyMajorMergerTime];

- `nodeMajorMergerTime` The time of the last major merger (as defined by the [nodeMajorMergerFraction] parameter) between this node and another, output as the time since the last major merger [mergingStatisticsNodeMajorMergerTime];

- `nodeFormationTime` The time at which the node is judged to have “formed”, defined as the time at which its main branch progenitor had a mass equal to a fraction [nodeFormationMassFraction] of the node’s initial mass in the merger tree [mergingStatisticsNodeFormationTime];

- `nodeHierarchyLevel` The depth of this node in the initial merger tree hierarchy. For example, a node on the main branch has level 0. A node which merges directly onto the main branch has level 1. A node that merges onto a node that merges directly onto the main branch has level 2, etc. [mergingStatisticsNodeHierarchyLevel];

**Initialization**

The times of the last mergers are stored each time a major merger occurs and this component is created (if necessary) the first time such a merger occurs. Formation times and hierarchy levels are computed during merger tree initialization.

**Differential Evolution**

The times of the last mergers do not evolve.
Event Evolution

Node mergers: The time of the last node merger in the parent node is reset to the current time if the merger is major.

Satellite merging: The time of the last merger is reset to the current time if the merger is major.

Node promotion: The time of the last major node merger is updated to that of the parent if that time is more recent.

12.14.2. “Recent” Implementation

Properties

The recent merging statistics implementation defines the following properties:

- `recentMajorMergerTime`: The number of node major mergers (defined as a merger with mass ratio in excess of `nodeMajorMergerFraction`) occurring within a recent interval, \( \Delta t \) where \( \Delta t = [\text{nodeRecentMajorMergerInterval}] \) if \( \text{nodeRecentMajorMergerIntervalType} = \text{absolute} \), or \( \Delta t = [\text{nodeRecentMajorMergerInterval}] \tau_{\text{dyn}} \) (where \( \tau_{\text{dyn}} \) is the halo dynamical time) if \( \text{nodeRecentMajorMergerIntervalType} = \text{dynamical} \).

Initialization

The number of recent mergers for each output time are initialized to zero.

Differential Evolution

The number of recent mergers do not evolve.

Event Evolution

Node mergers: If the merger is major and occurs within \( \Delta t \) of an output time, the number of recent major mergers is increased by one.

Satellite merging: N/A.

Node promotion: N/A.

12.15. Age Statistics

This component records statistics associated with galaxy or halo age.

12.15.1. “Standard” Implementation

Properties

The standard age statistics implementation defines the following properties:

- `(disk|spheroid)TimeWeightedIntegratedSFR`: A time-weighted integral over the star formation rate of the specified component `ageStatistics(Disk|Spheroid)TimeWeightedIntegratedSFR);`
12. Node Components

\( (\text{disk|spheroid})\text{IntegratedSFR} \) A unweighted integral over the star formation rate of the specified component \( \text{ageStatistics(Disk|Spheroid)IntegratedSFR} \);

**Initialization**

The integrals are all initialized to zero.

**Differential Evolution**

The time-weighted integrals evolve as:

\[
\frac{d(M_t)}{dt} = t\phi(t),
\]  

(12.83)

where \( t \) is cosmic time and \( \phi(t) \) is the star formation rate in the component. The unweighted integrals evolve as:

\[
\frac{dM_*}{dt} = \phi(t).
\]  

(12.84)

The mean cosmic time of formation for the stars in either component can be found by taking the ratio of time-weighted to unweighted integrals. The mean age can then be found by subtracting the mean cosmic time of formation from the present time.

**Event Evolution**

*Node mergers:* N/A.

*Satellite merging:* The values of the integrals are removed from the merging galaxy and transferred to the appropriate component of the target galaxy. Additionally, in a major merger, the values of the integral are transferred between components in the target galaxy as dictated by the merging rules.

*Node promotion:* N/A.

### 12.16. Formation Times

This component implements “formation times” of dark matter halos.

#### 12.16.1. “Cole2000” Implementation

**Properties**

The “Cole2000” formation times implementation defines the following properties:

*formationTime* The time at which the halo last “formed”. Formation is defined as an increase in the mass of the halo by a factor \([\text{haloReformationMassFactor}]\).

**Initialization**

The formation time is set to the current time and this component is created the first time such a merger occurs.
**Differential Evolution**

The formation time does not evolve. When the node mass exceeds the mass at the formation time by a factor \([\text{haloReformationMassFactor}]\) evolution is interrupted and the formation time reset to the current time.

**Event Evolution**

*Node mergers:* None.

*Satellite merging:* None.

*Node promotion:* None.

### 12.17. Indices

This component tracks various indices associated with nodes.

#### 12.17.1. “Standard” Implementation

**Properties**

The ‘standard’ indices implementation defines the following properties:

*branchTip* The index of the node at the tip of the branch to which the node belongs.

**Initialization**

The branch tip index is set to equal the index of each branch tip node in the tree.

**Differential Evolution**

The indices do not evolve.

**Event Evolution**

*Node mergers:* None.

*Satellite merging:* None.

*Node promotion:* None.

### 12.18. Inter-Output Quantities

This component tracks various quantities averaged between successive outputs.
12.18.1. “Standard” Implementation

Properties

The “standard” inter-output implementation defines the following properties:

**diskStarFormationRate** The mean star formation rate (in units of $M_\odot \text{ Gyr}^{-1}$) in the disk between the previous and current output times.

**spheroidStarFormationRate** The mean star formation rate (in units of $M_\odot \text{ Gyr}^{-1}$) in the spheroid between the previous and current output times.

Initialization

No initialization is carried out—component is created as needed.

Differential Evolution

The disk and spheroid star formation rates, $\langle \dot{M}_* \rangle$, evolve according to:

$$\frac{d}{dt} \langle \dot{M}_* \rangle = \frac{\dot{M}_*}{\Delta t},$$

where $\dot{M}_*$ is the instantaneous star formation rate in the corresponding component and $\Delta t$ is the time between the previous and next output times.

Event Evolution

**Node mergers:** None.

**Satellite merging:** The mean star formation rates from the satellite are added to those of the parent.

**Node promotion:** None.

**Output:** All quantities are zeroed after each output.
13. Physical Implementations

13.1. Accretion of Gas into Halos

The accretion rate of gas from the IGM into a dark matter halo is expected to depend on (at least) the rate at which that halo mass is growing, the depth of its potential well and the thermodynamical properties of the accreting gas. GALACTICUS implements the following calculations of gas accretion from the IGM, which can be selected via the `accretionHalosMethod` input parameter.

13.1.1. Simple Method

Selected using `accretionHalosMethod=simple`, this method sets the accretion rate of baryons into a halo to be:

\[
\dot{M}_{\text{accretion}} = \begin{cases} 
  (\Omega_b/\Omega_M)\dot{M}_{\text{halo}} & \text{if } V_{\text{virial}} > V_{\text{reionization}} \text{ or } z > z_{\text{reionization}} \\
  0 & \text{otherwise,}
\end{cases}
\]

(13.1)

where \(z_{\text{reionization}} = [\text{reionizationSuppressionRedshift}]\) is the redshift at which the Universe is reionized and \(V_{\text{reionization}} = [\text{reionizationSuppressionVelocity}]\) is the virial velocity below which accretion is suppressed after reionization. Setting \(V_{\text{reionization}}\) to zero will effectively switch off the effects of reionization on the accretion of baryonics. This algorithm attempts to offer a simple prescription for the effects of reionization and has been explored by multiple authors (e.g. Benson et al. 2002). In particular, Font et al. [2010] show that it produces results in good agreement with more elaborate treatments of reionization. For halos below the accretion threshold, any accretion rate that would have otherwise occurred is instead placed into the “failed” accretion rate. For halos which can accrete, and which have some mass in their “failed” reservoir, that mass will be added to the regular accretion rate at a rate equal to the mass of the “failed” reservoir times the specific growth rate of the halo. The gas accreted is assumed to be from a pristine IGM and so has zero abundances. Chemical abundances are computed from the chemical state functions (see §13.24).

Note that, if \(\dot{M}_{\text{halo}} < 0\) then negative accretion rates of gas into the node can result. This can be prevented by setting \([\text{accretionHalosSimpleNegativeAccretionAllowed}]=\text{false}\).

By default, gas is accreted whenever the halo is growing in total mass. However, setting \([\text{accretionHalosSimpleAccreteNewGrowthOnly}]=\text{true}\) causes accretion to occur only if the node mass is growing and exceeds the previous maximum node mass achieved along this branch of the merger tree. This requires use of a basic component which tracks the maximum mass along the branch (i.e. the `massMaximum` property).

13.1.2. Null Method

Selected using `accretionHalosMethod=null`, this method sets the accretion rate of baryons into a halo to be zero always.

13.2. Cosmology Functions

The background cosmology describes the evolution of an isotropic, homogeneous Universe within which our calculations are carried out. For the purposes of GALACTICUS, the background cosmology is used to
relate expansion factor/redshift to cosmic time and to compute the density of various components (e.g. dark matter, dark energy, etc.) at different epochs. Background cosmological models are specified via the `cosmologyMethod`, and the physics that must be implemented for each cosmological model is describe in more detail in §17.4.1. Currently implemented cosmological models are as follows.

### 13.2.1. Matter + Lambda

Selected with `cosmologyMethod=matter-lambda`, in this implementation cosmological relations are computed assuming a universe that contains only collisionless matter and a cosmological constant.

### 13.2.2. Matter + Dark Energy

Selected with `cosmologyMethod=matter-darkEnergy`, in this implementation cosmological relations are computed assuming a universe that contains only collisionless matter and dark energy with an equation of state $w(a) = w_0 + w_1 a (1 - a)$ [Jassal et al., 2005], with $w_0 = \text{[darkEnergyEquationOfStateW0]}$, and $w_1 = \text{[darkEnergyEquationOfStateW1]}$.

### 13.3. Circumnuclear Accretion Disks

Circumnuclear accretion disks surrounding supermassive black holes at the centers of galaxies influence the evolution of both the black hole (via accretion rates of mass and angular momentum and possibly by extracting rotational energy from the black hole) and the surrounding galaxy if they lead to energetic outflows (e.g. jets) from the nuclear region. Accretion disk type is specified via the `accretionDisksMethod`, and the physics that must be implemented for each accretion disk type is describe in more detail in §17.4.1. Current implementations of accretion disks are as follows.

#### 13.3.1. Shakura-Sunyaev Geometrically Thin, Radiatively Efficient Disks

Selected with `accretionDisksMethod=Shakura-Sunyaev`, this implementation assumes that accretion disks are always described by a radiatively efficient, geometrically thin accretion disk as described by Shakura and Sunyaev [1973]. The radiative efficiency of the flow is computed assuming that material falls into the black hole without further energy loss from the innermost stable circular orbit (ISCO), while the spin-up rate of the black hole is computed assuming that the material enters the black hole with the specific angular momentum of the ISCO (i.e. there are no torques on the material once it begins to fall in from the ISCO; Bardeen 1970). For these thin disks, jet power is computed, using the expressions from Meier (2001; his equations 4 and 5).

#### 13.3.2. Advection Dominated, Geometrically Thick, Radiatively Inefficient Flows (ADAFs)

Selected with `accretionDisksMethod=ADAF`, this implementation assumes that accretion is via an advection-dominated accretion flow (ADAF) [Narayan and Yi, 1994] which is radiatively inefficient and geometrically thick. The radiative efficiency of the flow, which will be zero for a pure ADAF, is controlled by `adafRadiativeEfficiencyType`. If set to `fixed`, then the radiative efficiency is set to the value of the input parameter `adafRadiativeEfficiency`. Alternatively, if set to `thinDisk` the radiative efficiency will be set to that of a Shakura-Sunyaev thin disk. The spin up rate of the black hole and the jet power produced as material accretes into the black hole are computed using the method of Benson and Babul [2009]. The maximum efficiency of the jet (in units of the accretion power $\dot{M}c^2$) is set by `adafJetEfficiencyMaximum`—in the model of Benson and Babul [2009] the jet efficiency diverges as
j → 1, setting a maximum is important to avoid numerical instabilities. The energy of the accreted material can be set equal to the energy at infinity (as expected for a pure ADAF) or the energy at the ISCO by use of the [adafEnergyOption] parameter (set to pureADAF or ISCO respectively). The ADAF structure is controlled by the adiabatic index, \( \gamma \), and viscosity parameter, \( \alpha \), which are specified via the [adafAdiabaticIndex] and [adafViscosityOption] input parameters respectively. The field-enhancing shear, \( g \), is computed using

\[
g = \exp(\omega \tau)
\]

if [adafFieldEnhanceType] is set to “exponential” where \( \omega \) is the frame-dragging frequency and \( \tau \) is the smaller of the radial inflow and azimuthal velocity timescales. If [adafFieldEnhanceType] is set to “linear” then the alternative version,

\[
g = 1 + \omega \tau
\]

is used instead. [adafViscosityOption] may be set to “fit”, in which case the fitting function for \( \alpha \) as a function of black hole spin is used:

\[
a(j) = 0.015 + 0.02j^4 \quad \text{if} \quad g = \exp(\omega \tau) \quad \text{and} \quad E = E_{\text{ISCO}},
\]

(13.2)

\[
a(j) = 0.025 + 0.08j^4 \quad \text{if} \quad g = 1 + \omega \tau \quad \text{and} \quad E = E_{\text{ISCO}},
\]

(13.3)

\[
a(j) = 0.010 + 0.00j^4 \quad \text{if} \quad g = \exp(\omega \tau) \quad \text{and} \quad E = 1,
\]

(13.4)

\[
a(j) = 0.025 + 0.02j^4 \quad \text{if} \quad g = 1 + \omega \tau \quad \text{and} \quad E = 1.
\]

(13.5)

### 13.3.3. “Switched” Disks

Selected with [accretionDisksMethod]=switched, this method allows for accretion disks to switched between radiatively efficient (Shakura-Sunyaev) and inefficient (ADAF) modes. The properties of the switched disk (e.g. radiative efficiency, jet power), are a linear combination of those of the Shakura-Sunyaev and ADAF modes, with the ADAF fraction being given by:

\[
f_{\text{ADAF}} = [1 + \exp(y_{\text{min}})]^{-1} + [1 + \exp(y_{\text{max}})]^{-1},
\]

(13.6)

where

\[
y_{\text{min}} = + \log(x/x_{\text{min}})/\Delta x,
\]

(13.7)

\[
y_{\text{max}} = - \log(x/x_{\text{max}})/\Delta x,
\]

(13.8)

and,

\[
x = \dot{M}/\dot{M}_{\text{Eddington}}.
\]

(13.9)

Here, \( x_{\text{min}} = [\text{accretionRateThinDiskMinimum}] \), \( x_{\text{max}} = [\text{accretionRateThinDiskMaximum}] \), and \( \Delta x = [\text{accretionRateTransitionWidth}] \). If either [accretionRateThinDiskMinimum] or [accretionRateThinDiskMaximum] is set to “none” then the corresponding term in eqn. (13.6) is excluded.

Additionally, if [accretionDiskSwitchedScaleAdafRadiativeEfficiency] is set to true then the radiative efficiency of the ADAF component is reduced by a factor \( x/x_{\text{min}} \) when \( x < x_{\text{min}} \).

### 13.3.4. Eddington-limited Disks

Selected with [accretionDisksMethod]=eddingtonLimited, this method does not assume any physical model for the accretion disk, but merely assumes that jets are powered at a fixed fraction \( [\text{accretionDiskJetPowerEddington}] \) of the Eddington luminosity. The radiative efficiency is similarly set at a fixed value of \( [\text{accretionDiskRadiativeEfficiencyEddington}] \). Since no physical model for the disk is assumed, the black hole spin up rate is always set to zero.

### 13.4. Dark Matter Structure Formation

A variety of functions are used to describe structure formation in dark matter dominated universes. These are described below.
13. Physical Implementations

13.4.1. Primordial Power Spectrum

The functional form of the primordial dark matter power spectrum is selected via the `powerSpectrumMethod` parameter. The power spectrum is computed from the specified primordial power spectrum and the transfer function (see §13.4.5) and normalized to a value of $\sigma_8$ specified by $[\text{sigma}_8]$.

(Running) Power Law Spectrum

Selected via `powerSpectrumMethod=powerLaw`, this method implements a primordial power spectrum of the form:

$$ P(k) \propto k^{n_{\text{eff}}(k)}, $$

where

$$ n_{\text{eff}}(k) = n_s + \frac{1}{2} \frac{dn}{d \ln k} \ln \left( \frac{k}{k_{\text{ref}}} \right), $$

where $n_s = \text{powerSpectrumIndex}$ is the power spectrum index at wavenumber $k_{\text{ref}} = \text{powerSpectrumReferenceWavenumber}$ and $dn/d \ln k = \text{powerSpectrumRunning}$ describes the running of this index with wavenumber.

13.4.2. Cosmological Mass Root Variance

The method used to compute the root variance of the cosmological mass field, $\sigma(M)$, is selected via the `cosmologicalMassVarianceMethod` parameter.

Filtered Power Spectrum

Selected via `cosmologicalMassVarianceMethod=filteredPowerSpectrum`, this method computes the mass root variance using:

$$ \sigma^2(M) = \frac{1}{2\pi^2} \int_0^\infty P(k)T^2(k)W^2(k)k^2 dk $$

where $P(k)$ is the primordial power spectrum (see §13.4.1), $T(k)$ is the transfer function (see §13.4.5), and $W(k)$ is the power spectrum variance window function (see §13.4.3).

13.4.3. Power Spectrum Variance Window Function

The functional form of the window function used in computing the variance of the power spectrum is selected via the `powerSpectrumWindowFunctionMethod` parameter. Note that when computing the normalization of the power spectrum to match the specified value of $\sigma_8$ a top-hat real-space window function is always used (as per the definition of $\sigma_8$).

Top-hat

Selected via `powerSpectrumWindowFunctionMethod=topHat`, this method implements a top-hat window function in real-space:

$$ W(k) = \frac{3(\sin(x) - x \cos(x))}{x^3}, $$

where $x = kR$ and $R = (3M/4\pi\bar{\rho})^{1/3}$ for a smoothing scale $M$ and mean matter density $\bar{\rho}$.
13.4. Dark Matter Structure Formation

Sharp in \( k \)-space

Selected via `powerSpectrumWindowFunctionMethod=kSpaceSharp`, this method implements a top-hat window function in \( k \)-space:

\[
W(k) = \begin{cases} 
1 & \text{if } k < k_s \\
0 & \text{if } k > k_s,
\end{cases}
\]

(13.14)

where if \([\text{powerSpectrumWindowFunctionSharpKSpaceNormalization}]=\text{natural}\) then \(k_s = (6\pi^2\bar{\rho}/M)^{1/3}\) for a smoothing scale \(M\) and mean matter density \(\bar{\rho}\). Otherwise, \([\text{powerSpectrumWindowFunctionSharpKSpaceNormalization}]=\text{natural}\) must be set to a numerical value, \(\alpha\), in which case \(k_s = \alpha/R_{th}\) with \(R_{th} = 3M/4\pi\bar{\rho}\) for a smoothing scale \(M\) and mean matter density \(\bar{\rho}\).

Hybrid of Top-hat and Sharp in \( k \)-space

Selected via `powerSpectrumWindowFunctionMethod=topHatKSpaceSharpHybrid`, this method implements a convolution of a top-hat window function and sharp \( k \)-space window function in \( k \)-space:

\[
W(k) = W_{th}(k)W_s(k),
\]

(13.15)

where

\[
W(k) = \frac{3(\sin(x) - x \cos(x))}{x^3},
\]

(13.16)

where \(x = kR_{th}\), and

\[
W_s(k) = \begin{cases} 
1 & \text{if } k < k_s \\
0 & \text{if } k > k_s,
\end{cases}
\]

(13.17)

where \(k_s = \alpha/R_s\) if \([\text{powerSpectrumWindowFunctionSharpKSpaceNormalization}]=\text{natural}\) is assigned a numerical value. Alternatively, if \([\text{powerSpectrumWindowFunctionSharpKSpaceNormalization}]=\text{natural}\) then the value of \(\alpha\) is chosen such that \(k_s = (6\pi^2\bar{\rho}/M)^{1/3}\) if \(R_s = 3M/4\pi\bar{\rho}\).

The radii, \(R_{th}\) and \(R_s\), are chosen such that:

\[
R_{th}^2 + R_s^2 = (3M/4\pi\bar{\rho})^{2/3}
\]

(13.18)

\[
R_s = \beta R_{th},
\]

(13.19)

where \(\beta = [\text{powerSpectrumWindowFunctionSharpKSpaceTopHatRadiiRatio}]\).

13.4.4. Non-linear Matter Power Spectrum

The non-linear matter power spectrum method is selected via the `powerSpectrumNonlinearMethod` parameter.

Linear

Selected via `powerSpectrumNonlinearMethod=linear`, this method simply returns the linear matter power spectrum. It is intended primarily for testing purposes.

Peacock & Dodds (1996)

Selected via `powerSpectrumNonlinearMethod=Peacock-Dodds1996`, this method uses the fitting function of Peacock and Dodds [1996] to compute the non-linear matter power spectrum.
13. Physical Implementations

**CosmicEmu**

Selected via `powerspectrumNonlinearMethod=CosmicEmu`, this method uses the cosmic emulator ("CosmicEmu") code of Lawrence et al. [2010] to evaluate the non-linear matter power spectrum. The CosmicEmu code will be downloaded, compiled and run as necessary if this option is utilized.

### 13.4.5. Transfer Function

The functional form of the cold dark matter transfer function is selected via the `transferFunctionMethod` parameter. The power spectrum is computed from the specified transfer function and the primordial power spectrum (see §13.4.1) and normalized to a value of $\sigma_8$ specified by $[\sigma_8]$.

**Null**

Selected with `transferFunctionMethod=null`, this method assumes $T(k) = 1$ for all $k$.

**BBKS**

Selected with `transferFunctionMethod=BBKS`, this method uses the fitting function of Bardeen et al. [1986] to compute the CDM transfer function. The BBKS warm dark matter transfer function can be used by specifying the appropriate streaming length (in Mpc) via the `[transferFunctionWDMFreeStreamingLength]` parameter.

**Eisenstein & Hu**

Selected with `transferFunctionMethod=Eisenstein-Hu1999`, this method uses the fitting function of Eisenstein and Hu [1999] to compute the CDM transfer function. It requires that the effective number of neutrino species be specified via the `effectiveNumberNeutrinos` parameter and summed mass of all neutrino species (in eV) be specified via the `summedNeutrinoMasses` parameter. Additionally, the transfer function can be modified to model warm dark matter using the fitting function given by Barkana et al. [2001]:

$$T(k) \rightarrow T(k)(1 + [\epsilon k R_0^c]^{2\nu})^{-\eta/\nu},$$

where $R_0^c = [transferFunctionWdmCutOffScale]$, $\epsilon = [transferFunctionWdmEpsilon]$, $\eta = [transferFunctionWdmEta]$, $\nu = [transferFunctionWdmNu]$.

**CAMB**

Selected with `transferFunctionMethod=CAMB`, this method uses the CAMB code to compute the CDM transfer function. It requires that the mass fraction of helium in the early Universe be specified via the $Y_{He}$ parameter. CAMB will be downloaded and run if the transfer function needs to be computed. It will then be stored in a file for future reference.

**File**

Selected with `transferFunctionMethod=file`, this method reads a tabulated transfer function from an XML file (specified via the `transferFunctionFile` parameter), interpolating between tabulated points. The structure of the transfer function file is described in §17.4.1.

### 13.4.6. Linear Growth Function

The function describing the amplitude of linear perturbations is selected via the `linearGrowthMethod` parameter.
Simple

Selected with `linearGrowthMethod=simple`, this method calculates the growth of linear perturbations using standard perturbation theory in a Universe consisting of matter and a cosmological constant. Perturbations in the baryons are treated just as for dark matter (i.e. pressure forces are ignored), while perturbations in the radiation are assumed not to grow.

13.4.7. Critical Overdensity

The method used to compute the critical linear overdensity at which overdense regions virialize is selected via the `criticalOverdensityMethod` parameter.

Spherical Collapse (Matter + Cosmological Constant)

Selected with `criticalOverdensityMethod=sphericalTopHat` this method calculates critical overdensity using a spherical top-hat collapse model assuming a Universe which contains matter and a cosmological constant (see, for example, Percival 2005).

Kitayama & Suto (1996)

Selected with `criticalOverdensityMethod=Kitayama-Suto1996` this method calculates critical overdensity using the fitting formula of Kitayama and Suto [1996], and so is valid only in flat cosmological models (an error will be reported in non-flat models). Specifically,

\[
\delta_{\text{crit}}(t) = \frac{3(12\pi)^{2/3}}{20} \left[ 1 + 0.0123 \log_{10} \left( \Omega_{\text{matter}}(t) \right) \right] / D(t). \tag{13.21}
\]

13.4.8. Critical Overdensity Mass Scaling

The method used to compute the scaling with mass of the critical linear overdensity at which overdense regions virialize is selected via the `criticalOverdensityMassScalingMethod` parameter.

Null

Selected with `criticalOverdensityMassScalingMethod=null` this method assumes that the critical overdensity is independent of mass.

Warm Dark Matter

Selected with `criticalOverdensityMassScalingMethod=warmDarkMatter` this method assumes that the critical overdensity scales with mass as expected for warm dark matter using the results of Barkana et al. [2001]. Specifically, the critical overdensity is multiplied by a factor

\[
\exp \left[ \left( \frac{M_J}{8M} \right)^{1.40} + \left( \frac{M_J}{8M} \right)^{0.45} \right], \tag{13.22}
\]

where \( M \) is the mass in question, \( M_J \) is the effective Jeans mass of the warm dark matter as defined by Barkana et al. [2001; their eqn. 10]:

\[
M_J = 3.06 \times 10^8 \left( \frac{1 + \frac{z_{eq}}{3000}}{0.15} \right)^{1.5} \left( \frac{\Omega M h^2}{0.15} \right)^{1/2} (\frac{9X}{1.5})^{-1} (m_X/1.0\text{keV})^{-4}, \tag{13.23}
\]
13. Physical Implementations

the redshift of matter-radiation equality is given by

$$z_{eq} = 3600 \left( \frac{\Omega_M h^2_0}{0.15} \right) - 1,$$

(13.24)

and $g_X$ and $m_X$ are the effective number of degrees of freedom and the mass of the warm dark matter particle respectively. This fitting function has been found the fit the numerical results of Barkana et al. [2001] well.

13.4.9. Virial Density Contrast

The method used to compute the mean density contrast of virialized dark matter halos is selected via the `virialDensityContrastMethod` parameter.

**Bryan & Norman (1998) Fitting Function**

Selected with `virialDensityContrastMethod=Bryan-Norman1998` this method calculates virial density contrast using the fitting functions given by Bryan and Norman [1998]. As such, it is valid only for $\Omega_\Lambda = 0$ or $\Omega_M + \Omega_\Lambda = 1$ cosmologies and will abort on other cosmologies.

**Fixed**

Selected with `virialDensityContrastMethod=fixed` this method uses a fixed virial density contrast of $[\text{virialDensityContrastFixed}]$, defined relative to `criticalDensity` and `meanDensity` as specified by $[\text{virialDensityContrastFixedType}]$.

**Spherical Collapse (Matter + Cosmological Constant)**

Selected with `virialDensityContrastMethod=sphericalTopHat` this method calculates virial density contrast using a spherical top-hat collapse model assuming a Universe which contains matter and a cosmological constant (see, for example, Percival 2005).

**Kitayama & Suto (1996)**

Selected with `virialDensityContrastMethod=Kitayama-Suto1996` this method calculates virial density contrast using the fitting formula of Kitayama and Suto [1996], and so is valid only in flat cosmological models (an error will be reported in non-flat models). Specifically,

$$\Delta_{\text{virial}}(t) = 18\pi^2 [1 + 0.4093 \left\{ \frac{1}{\Omega_{\text{matter}}(t)} - 1 \right\} (t)^{0.9052}].$$

(13.25)

**Friends-of-Friends**

Selected with `virialDensityContrastMethod=friendsofFriends` this method computes the virial density contrast consistent with a given friends-of-friends algorithm linking length. According to Lacey and Cole [1994], the friends-of-friends algorithm selects objects bounded by an isodensity contour of density contrast

$$\Delta_{\text{iso}} = \frac{3}{2\pi b^3},$$

(13.26)

where $b = [\text{virialDensityContrastFoFLinkingLength}]$ is the dimensionless linking length of the algorithm (i.e. the linking length in units of the mean interparticle spacing). The virial density contrast is then given by:

$$\Delta_{\text{vir}} = \frac{\bar{\rho}_{\text{vir}}}{\rho(\bar{r}_{\text{vir}})} \Delta_{\text{iso}},$$

(13.27)
where $\bar{\rho}_{\text{vir}}$ is the mean density inside the virial radius and $\rho(r_{\text{vir}})$ is the density at the virial radius. The ratio $\bar{\rho}_{\text{vir}}/\rho(r_{\text{vir}})$ is specified via the parameter $[\text{virialDensityContrastForDensityRatio}]$. Its default value of 4.688 is appropriate for an NFW halo of concentration $c = 6.88$ which is the concentration found by Prada et al. [2011] for halos with $\sigma = 1.686$ which is the approximate critical overdensity for collapse).

13.4.10. Halo Bias

The dark matter halo linear bias method is selected via the `darkMatterHaloBiasMethod` parameter.

**Press-Schechter**

Selected with `darkMatterHaloBiasMethod=Press-Schechter` this method uses a bias consistent with the halo mass function of Press and Schechter [1974] (see [Mo and White, 1996]).

**Sheth-Tormen**

Selected with `darkMatterHaloBiasMethod=SMT` this method uses a bias consistent with the halo mass function of Sheth et al. [2001].

**Tinker**

Selected with `darkMatterHaloBiasMethod=Tinker2010` this method uses the functional form proposed by Tinker et al. [2010] to compute the halo bias. The bias is computed at the appropriate virial overdensity (see §13.4.9).

13.4.11. Halo Mass Function

The dark matter halo mass function (i.e. the number of halos per unit volume per unit mass interval) is selected via the `haloMassFunctionMethod` parameter.

**Press-Schechter**

Selected with `haloMassFunctionMethod=Press-Schechter` this method uses the functional form proposed by Press and Schechter [1974] to compute the halo mass function.

**Sheth-Tormen**

Selected with `haloMassFunctionMethod=Sheth-Tormen` this method uses the functional form proposed by Sheth et al. [2001] to compute the halo mass function. Specifically,

$$n(M,t) = 2\Omega M \rho_{\text{crit}} \sigma^2(M) f[S(M,t)],$$

(13.28)

where $\alpha = d\ln \sigma/d\ln M$ and $f[S]$ is the excursion set barrier first crossing distribution for variance $S(M) = \sigma^2(M)$, computed using the selected `excursionSetFirstCrossingMethod` (see §17.4.1).

**Tinker**

Selected with `haloMassFunctionMethod=Tinker2008` this method uses the functional form proposed by Tinker et al. [2008] to compute the halo mass function. The mass function is computed at the appropriate virial overdensity (see §13.4.9).
13.5. Conditional Mass Functions

The method to be used for computing the conditional mass function (for use in data modelling) is specified by the conditionalMassFunctionMethod parameter.

13.5.1. Behroozi et al. (2010)

Selected with conditionalMassFunctionMethod=Behroozi2010, this method uses the functional form given by Behroozi et al. (2010; see also Leauthaud et al. 2011). The mean number of central galaxies of stellar mass, $M_\star$, in halos of mass $M_h$ is given by:

$$\log_{10} \left[ f_{SHMR}^{-1}(M_\star) \right] = \log_{10}(M_h) = \log_{10}(M_1) + \beta \log_{10} \left( \frac{M_\star}{M_{\star,0}} \right) + \frac{(M_\star/M_{\star,0})^\delta}{1 + (M_\star/M_{\star,0})^{-\gamma}} - \frac{1}{2},$$

(13.29)

where

$$\langle n_{cen}(M_h|M_\star) \rangle = \frac{1}{2} \left[ 1 - \text{erf} \left( \frac{\log_{10}(M_\star) - \log_{10}[f_{SHMR}(M_h)]}{\sqrt{2} \sigma_{\log M_\star}} \right) \right].$$

(13.30)

The mean number of satellite galaxies of stellar mass, $M_\star$, in halos of mass $M_h$ is given by:

$$\langle N_{sat}(M_h|M_\star) \rangle = \langle n_{cen}(M_h|M_\star) \rangle \left( \frac{f_{SHMR}^{-1}(M_\star)}{M_{sat}} \right)^{\alpha_{sat}} \exp \left( -\frac{M_{cut}}{f_{SHMR}(M_\star)} \right),$$

(13.31)

where

$$\frac{M_{sat}}{10^{12} M_\odot} = B_{sat} \left( \frac{f_{SHMR}^{-1}(M_\star)}{10^{12} M_\odot} \right)^{\beta_{sat}},$$

(13.32)

and

$$\frac{M_{cut}}{10^{12} M_\odot} = B_{cut} \left( \frac{f_{SHMR}^{-1}(M_\star)}{10^{12} M_\odot} \right)^{\beta_{cut}}.$$

(13.33)

The number of centrals in a given halo is assumed to be Bernoulli distributed (0 or 1), while the number of satellites is assumed to be Poisson distributed with the above mean. The parameters of the distribution are controlled by the input parameters as listed in Table 13.1.

13.6. Cooling of Gas Inside Halos

The cooling of gas within dark matter halos is controlled by a number of different algorithms which will be described below.

13.6.1. Cooling Function

The cooling function of gas, $\Lambda(\rho, T, Z)$, is implemented by the algorithm(s) selected using the coolingFunctionMethods parameter. If more than one cooling function is specified, then the net cooling function is a sum over all of those selected.

Atomic Collisional Ionization Equilibrium Using CLOUDY

Selected using coolingFunctionMethods=atomic_CIE_Cloudy, this method computes the cooling function using the CLOUDY code and under the assumption of collisional ionization equilibrium with no molecular contribution. Abundances are Solar, except for zero metallicity calculations which use CLOUDY’s “primordial” metallicity. The helium abundance for non-zero metallicity is scaled between primordial and...
13.6. Cooling of Gas Inside Halos

Table 13.1.: The correspondence between parameters of the Behroozi et al. [2010] model for the conditional mass function and GALACTICUS input parameters.

<table>
<thead>
<tr>
<th>Model parameter</th>
<th>Input parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_{sat}$</td>
<td>conditionalMassFunctionBehrooziAlphaSatellite</td>
</tr>
<tr>
<td>$\log_{10}(M_1/M_\odot)$</td>
<td>conditionalMassFunctionBehrooziLog10M1</td>
</tr>
<tr>
<td>$\log_{10}(M_{*,0}/M_\odot)$</td>
<td>conditionalMassFunctionBehrooziLog10Mstar0</td>
</tr>
<tr>
<td>$\beta$</td>
<td>conditionalMassFunctionBehrooziBeta</td>
</tr>
<tr>
<td>$\delta$</td>
<td>conditionalMassFunctionBehrooziDelta</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>conditionalMassFunctionBehrooziGamma</td>
</tr>
<tr>
<td>$\sigma_{log M_*}$</td>
<td>conditionalMassFunctionBehrooziSigmaLogMstar</td>
</tr>
<tr>
<td>$B_{cut}$</td>
<td>conditionalMassFunctionBehrooziBCut</td>
</tr>
<tr>
<td>$B_{sat}$</td>
<td>conditionalMassFunctionBehrooziBSatellite</td>
</tr>
<tr>
<td>$\beta_{cut}$</td>
<td>conditionalMassFunctionBehrooziBetaCut</td>
</tr>
<tr>
<td>$\beta_{sat}$</td>
<td>conditionalMassFunctionBehrooziBetaSatellite</td>
</tr>
</tbody>
</table>

Solar values linearly with metallicity. The CLOUDY code will be downloaded and run to compute the cooling function as needed, which will then be stored for future use. As this process is slow, a precomputed table is provided with GALACTICUS. If metallicities outside the range tabulated in this file are required it will be regenerated with an appropriate range.

**Collisional Ionization Equilibrium From File**

Selected using `coolingFunctionMethods=CIE_from_file`, in this method the cooling function is read from a file specified by the `coolingFunctionFile` parameter. The format of this file is specified in §17.4.1. The cooling function is assumed to be computed under conditions of collisional ionization equilibrium and therefore to scale as $\rho^2$.

**CMB Compton Cooling**

Selected using `coolingFunctionMethods=CMB_Compton`, this method computes the cooling function due to Compton scattering off of CMB photons:

$$
\Lambda = \frac{4\sigma_T a k_B n_e}{m_e c} T_{CMB}^4 (T - T_{CMB}),
$$

where $\sigma_T$ is the Thompson cross-section, $a$ is the radiation constant, $k_B$ is Boltzmann’s constant, $n_e$ is the number density of electrons, $m_e$ is the electron mass, $c$ is the speed of light, $T_{CMB}$ is the CMB temperature at the current cosmic epoch and $T$ is the temperature of the gas. The electron density is computed from the selected chemical state method (see §13.24).

**Molecular Hydrogen (Galli-Palla)**

Selected using `coolingFunctionMethods=molecularHydrogenGalliPalla`, this method computes the cooling function due to molecular hydrogen using the results of Galli and Palla [1998]. For the $\text{H}^-\text{H}_2$ cooling function, the fitting functions from Galli and Palla [1998] are used. For the $\text{H}^+_2-e^-$ and $\text{H}--\text{H}_2^+$ cooling functions fitting functions to the results plotted in Suchkov and Shchekinov [1978] are used:

$$
\log_{10} \left( \frac{\Lambda(T)}{\text{erg s}^{-1}\text{cm}^3} \right) = C_0 + C_1 \log_{10} \left( \frac{T}{K} \right) + C_2 \left[ \log_{10} \left( \frac{T}{K} \right) \right]^2,
$$
Table 13.2.: Coefficients of $H^+_2$ cooling functions as appearing in the fitting function, eq. 13.35.

<table>
<thead>
<tr>
<th>Interaction</th>
<th>$C_0$</th>
<th>$C_1$</th>
<th>$C_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H^+_2 e^-$</td>
<td>-33.33</td>
<td>5.565</td>
<td>-0.4675</td>
</tr>
<tr>
<td>$H^- H^+_2$</td>
<td>-35.28</td>
<td>5.862</td>
<td>-0.5124</td>
</tr>
</tbody>
</table>

where the coefficients $C_{0-2}$ are given in Table 13.2.

13.6.2. Cooling Rate

The algorithm used to compute the rate at which gas drops out of the hot halo due to cooling is selected with the coolingRateMethod parameter.

**Cole et al. (2000)**

Selected with coolingRateMethod=Cole2000, this method computes the cooling rate using the algorithm of Cole et al. [2000]. The cooling rate is given by

$$
\dot{M}_{\text{cool}} = \begin{cases} 
4\pi r_{\text{infall}}^2 \rho(r_{\text{infall}}) \dot{r}_{\text{infall}} & \text{if } r_{\text{infall}} < r_{\text{hot,outer}} \\
0 & \text{if } r_{\text{infall}} \geq r_{\text{hot,outer}},
\end{cases}
$$

(13.36)

where $\rho(r)$ is the density profile of the hot halo, and $r_{\text{infall}}$ is the infall radius (see §13.6.6). The cooling rate is also set to zero in halos with virial velocities below $[\text{coolingCutOffVelocity}]$ at redshifts below $[\text{coolingCutOffRedshift}]$.

**Simple**

Selected with coolingRateMethod=simple, this method computes the cooling rate using

$$
\dot{M}_{\text{cool}} = \frac{M_{\text{hot}}}{\tau_{\text{cool}}},
$$

(13.37)

where $\tau_{\text{cool}} = [\text{coolingRateSimpleTimescale}]$.

**Simple Scaling**

Selected with coolingRateMethod=simpleScaling, this method computes the cooling rate using

$$
\dot{M}_{\text{cool}} = \frac{M_{\text{hot}}}{\tau_{\text{cool}}(M_{\text{halo}}, z)},
$$

(13.38)

where

$$
\tau_{\text{cool}} = \tau_{\text{cool,0}}(1 + z)^{\beta_{\text{cool}}} \exp \left( \left[ \frac{M_{\text{halo}}}{M_{\text{transition}}} \right]^{\gamma_{\text{cool}}} \right),
$$

(13.39)

$\tau_{\text{cool,0}} = [\text{coolingRateSimpleScalingTimescale}]$, $\beta_{\text{cool}} = [\text{coolingRateSimpleScalingTimescaleExponent}]$, $M_{\text{transition}} = [\text{coolingRateSimpleScalingTransitionMass}]$, and $\gamma_{\text{cool}} = [\text{coolingRateSimpleScalingCutoffExponent}]$. 

252
White & Frenk

Selected with `coolingRateMethod=White-Frenk1991`, this method computes the cooling rate using the expression given by White and Frenk [1991], namely

\[
\dot{M}_{\text{cool}} = \begin{cases} 
4\pi r_{\text{infall}}^2 \rho(r_{\text{infall}}) r_{\text{infall}} & \text{if } r_{\text{infall}} < r_{\text{hot,outer}} \\
M_{\text{hot}} / \tau_{\text{halo,dynamical}} & \text{if } r_{\text{infall}} \geq r_{\text{hot,outer}}, 
\end{cases}
\]

(13.40)

where \( r_{\text{infall}} \) is the infall radius (see §13.6.6) in the hot halo and \( \rho(r) \) is the density profile of the hot halo.

13.6.3. Cooling Rate Modifier

Algorithms that modify the rate of cooling of gas are specified via the `coolingRateModifierMethod` directive.

Cut-Off

This method sets the cooling rate to zero in halos with virial velocities below \([coolingCutOffVelocity]\) at redshifts below/above \([coolingCutOffRedshift]\) for \([coolingCutOffWhen]=after/before\). In other halos the cooling rate is not modified.

13.6.4. Cooling Radius

The algorithm used to compute the cooling radius is selected via the `coolingRadiusMethod` parameter.

Simple

Selected with `coolingRadiusMethod=simple`, this method computes the cooling radius by seeking the radius at which the time available for cooling (see §13.6.9) equals the cooling time (see §13.6.8). The growth rate is determined consistently based on the slope of the density profile, the density dependence of the cooling function and the rate at which the time available for cooling is increasing. This method assumes that the cooling time is a monotonic function of radius.

Isothermal

Selected with `coolingRadiusMethod=isothermal`, this method computes the cooling radius by assuming an isothermal density profile, and a cooling rate proportional to density squared. This implies a cooling time:

\[
t_{\text{cool}} \equiv \frac{E}{\dot{E}} \propto \rho(r)^{-1}.
\]

(13.41)

The cooling radius is then derived using

\[
\rho(r_{\text{cool}}) \propto t^{-1}_{\text{available}}
\]

(13.42)

which implies

\[
r_{\text{cool}} = r_{\text{virial}} \left( \frac{t_{\text{available}}}{t_{\text{cool,virial}}} \right)^{1/2},
\]

(13.43)

where \( t_{\text{cool,virial}} \) is the cooling time at the virial radius.
13. Physical Implementations

13.6.5. Cooling: Freefall Radius

The algorithm used to compute the freefall radius for cooling is selected via the \texttt{freefallRadiusMethod} parameter.

**Dark Matter Halo**

Selected with \texttt{freefallRadiusMethod=darkMatterHalo}, this method assumes that the freefall radius corresponds to the radius at which the freefall time in the dark matter halo equals the time available for freefall (see §13.6.10).

13.6.6. Cooling: Infall Radius

The algorithm used to compute the infall radius for cooling is selected via the \texttt{infallRadiusMethod} parameter.

**Cooling Radius**

Selected with \texttt{infallRadiusMethod=coolingRadius}, this method assumes that the infall radius equals the cooling radius (see §13.6.4).

**Cooling and Freefall Radii**

Selected with \texttt{infallRadiusMethod=cooling and freefall}, this method assumes that the infall radius is equal to the smaller of the cooling and freefall radii (see §13.6.4 and §13.6.5).

**Dark Matter Halo**

Selected with \texttt{freefallRadiusMethod=darkMatterHalo}, this method computes the freefall radius by finding the radius in the dark matter halo profile from which a test particle could have free-fallen to zero radius (assuming it began at rest) in the time available for freefall (see §13.6.10).

13.6.7. Cooling Specific Angular Momentum

The algorithm used to compute the specific angular momentum of cooling gas is selected via the \texttt{coolingSpecificAngularMomentumMethod} parameter.

**Constant Rotation**

Selected with \texttt{coolingSpecificAngularMomentumMethod=constantRotation}, this implementation assumes that the specific angular momentum of cooling gas is given either by

\[
j_{\text{cool}} = \langle j \rangle r_{\text{cool}} A, \tag{13.44}\]

where \( r_{\text{cool}} \) is the cooling radius, \( A \) is the rotation normalization (see ) and \( \langle j \rangle \) is the mean specific angular momentum of the cooling gas, if \texttt{[coolingAngularMomentumUseInteriorMean]=false}, or by

\[
j_{\text{cool}} = \langle j \rangle I_n(r_{\text{cool}})/I_2(r_{\text{cool}}) A, \tag{13.45}\]

where \( I_n(r) \) is the \( n^{th} \) radial moment of the hot gas density profile from 0 to \( r \) (this therefore gives the mean specific angular momentum interior to radius \( r \)), if \texttt{[coolingAngularMomentumUseInteriorMean]=true}.

If \texttt{[coolingMeanAngularMomentumFrom]=darkMatter} then \( \langle j \rangle \) is the mean specific angular momentum of the dark matter halo, computed from its spin parameter, while if \texttt{[coolingMeanAngularMomentumFrom]=hotGas}
then \( \langle j \rangle \) is equal to the mean specific angular momentum of gas currently in the hot gas reservoir. If \([\text{coolingRotationVelocityFrom}] = \text{darkMatter}\) then the rotation normalization \( A \) is computed using the dark matter density profile, while if \([\text{coolingRotationVelocityFrom}] = \text{hotGas}\) it is computed using the density profile of the hot gas reservoir.

**Mean**

Selected with \texttt{coolingSpecificAngularMomentumMethod=mean}, this assumes that the specific angular momentum of cooling gas is given by

\[
j_{cool} = J_{hot}/M_{hot},
\]

where \( J_{hot} \) and \( M_{hot} \) are the total angular momentum and mass of the hot halo respectively.

**13.6.8. Cooling Time**

The algorithm used to compute the time taken for gas to cool (i.e. the cooling time) is selected via the \texttt{coolingTimeMethod} parameter.

**Simple**

Selected with \texttt{coolingTimeMethod=simple}, this method assumes that the cooling time is simply

\[
t_{cool} = \frac{N k_B T n_{tot}}{2 \Lambda},
\]

where \( N = \text{coolingTimeSimpleDegreesOfFreedom} \) is the number of degrees of freedom in the cooling gas which has temperature \( T \) and total particle number density (including electrons) \( n_{tot} \) and \( \Lambda \) is the cooling function.

**13.6.9. Time Available for Cooling**

The method used to determine the time available for cooling (i.e. the time for which gas in a halo has been able to cool) is selected by the \texttt{coolingTimeAvailableMethod} parameter.

**Halo Formation**

Selected with \texttt{coolingTimeAvailableMethod=haloFormation}, this method assumes that the time available for cooling is equal to

\[
t_{available} = t - t_{form},
\]

where \( t_{form} \) is the time at which the halo formed (see §12.16).

**White & Frenk (1991)**

Selected with \texttt{coolingTimeAvailableMethod=White-Frenk1991}, this method assumes that the time available for cooling is equal to

\[
t_{available} = \exp [f \ln t_{Universe} + (1 - f) \ln t_{dynamical}],
\]

where \( f = \text{coolingTimeAvailableAgeFactor} \) is an interpolating factor, \( t_{Universe} \) is the age of the Universe and \( t_{dynamical} \) is the dynamical time in the halo. The original White and Frenk [1991] algorithm corresponds to \( f = 1 \).
13. Physical Implementations

13.6.10. Time Available for Freefall During Cooling

The method used to determine the time available for freefall during cooling calculations (i.e. the time for which gas in a halo has been able to freefall) is selected by the `freefallTimeAvailableMethod` parameter.

**Halo Formation**

Selected with `freefallTimeAvailableMethod=haloFormation`, this method assumes that the time available for freefall is equal to

\[ t_{\text{available}} = t - t_{\text{form}}, \]

(13.50)

where \( t_{\text{form}} \) is the time at which the halo formed (see §12.16).

13.7. Cosmology

The method used to compute cosmological relations (e.g. expansion factor as a function of time) is selected by the `cosmologyMethod` parameter.

13.7.1. Matter + Cosmological Constant Universes

Selected with `cosmologyMethod=matter-lambda`, this method assumes a universe which contains only matter and a cosmological constant.

13.8. Dark Matter Halos

Several algorithms are used to implement dark matter halos.

13.8.1. Mass Accretion History

The method used to compute mass accretion histories of dark matter halos is selected via the `darkMatterAccretionHistoryMethod` parameter.

**Wechsler et al. (2002)**

Selected with `darkMatterAccretionHistoryMethod=Wechsler2002`, under this method the mass accretion history is given by [Wechsler et al., 2002]:

\[ M(t) = M(t_0) \exp \left( -2a_c \left[ \frac{a(t_0)}{a(t)} - 1 \right] \right), \]

(13.51)

where \( t_0 \) is some reference time and \( a_c \) is a characteristic expansion factor defined by Wechsler et al. [2002] to correspond to the formation time of the halo (using the formation time definition of Bullock et al. 2001).

**Zhao et al. (2009)**

Selected with `darkMatterAccretionHistoryMethod=Zhao2009`, under this method the algorithm given by Zhao et al. [2009] to compute mass accretion histories. In particular, Zhao et al. [2009] give a fitting
function for the quantity \( \frac{d \ln \sigma(M)}{d \ln \delta_c(t)} \) for the dimensionless growth rate in a mass accretion history at time \( t \) and halo mass \( M \). This is converted to a dimensionful growth rate using

\[
\frac{dM}{dt} = \left( \frac{d \ln \sigma(M)}{d \ln M} \right)^{-1} \left( \frac{M}{\delta_c(t)} \right) \left( \frac{d \ln \sigma(M)}{d \ln \delta_c(t)} \right).
\] (13.52)

This differential equation is then solved numerically to find the mass accretion history.

### 13.8.2. Density Profile

The method uses to compute density profiles of dark matter halos is selected via the \texttt{darkMatterProfileMethod} parameter.

**Isothermal**

Selected with \texttt{darkMatterProfileMethod=isothermal}, under this method the density profile is given by:

\[
\rho_{dark\text{matter}}(r) \propto r^{-2},
\] (13.53)

normalized such that the total mass of the node is enclosed with the virial radius.

**NFW**

Selected with \texttt{darkMatterProfileMethod=NFW}, under this method the NFW density profile [Navarro et al., 1997] is used

\[
\rho_{dark\text{matter}}(r) \propto \left( \frac{r}{r_s} \right)^{-1} \left[ 1 + \left( \frac{r}{r_s} \right) \right]^{-2},
\] (13.54)

normalized such that the total mass of the node is enclosed with the virial radius and with the scale length \( r_s = r_{\text{virial}}/c \) where \( c \) is the halo concentration (see §13.8.3).

**Einasto**

Selected with \texttt{darkMatterProfileMethod=Einasto}, under this method the Einasto density profile (e.g. Cardone et al. 2005) is used

\[
\rho_{dark\text{matter}}(r) = \rho_{-2} \exp \left( -\frac{\alpha}{2} \left[ \left( \frac{r}{r_s} \right)^\alpha - 1 \right] \right),
\] (13.55)

normalized such that the total mass of the node is enclosed with the virial radius and with the characteristic length \( r_{-2} = r_{\text{virial}}/c \) where \( c \) is the halo concentration (see §13.8.3). The shape parameter, \( \alpha \), is set using the density profile shape method (see §13.8.4).

### 13.8.3. Dark Matter Halo Profile Concentration

The method uses to compute the concentrations of dark matter profiles is selected via the \texttt{darkMatterProfileConcentrationMethod} parameter.
Navarro, Frenk & White (1996)

Selected with `darkMatterProfileConcentrationMethod=NFW1996`, under this method the concentration is computed using the algorithm from Navarro et al. [1996]. In this algorithm, for a given halo of mass $M$ at time $t_0$, a formation time is defined as the epoch at which there is a 50% probability (according to extended Press-Schechter theory) for a progenitor halo to have a mass greater than $fM$, where $f=[\text{nfw96ConcentrationF}]$ is a parameter of the algorithm. This implies formation when the critical overdensity for collapse is

$$
\delta_{\text{crit}}(t_{\text{form}}) = \left[ 2\nu_{1/2}^2 \left\{ \sigma(fM)^2 - \sigma(M)^2 \right\} \right]^{1/2} + \delta_{\text{crit}}(t_0),
$$

(13.56)

where $\nu_{1/2} = [\text{erfc}^{-1}(1/2)]^{1/2}$. Navarro et al. [1996] then assume an overdensity at collapse of

$$
\Delta(t_{\text{form}}) = C \left[ \frac{a(t_0)}{a(t_{\text{form}})} \right]^3
$$

(13.57)

where $C=[\text{nfw96ConcentrationC}]$ is a parameter of the algorithm. The concentration is then determined by solving

$$
\frac{\Delta(t_{\text{form}})}{\Delta_{\text{virial}}(t_0)} = \frac{c^3}{3\ln(1+c) - c/(1+c)}.
$$

(13.58)

Gao (2008)

Selected with `darkMatterProfileConcentrationMethod=gao2008`, under this method the concentration is computed using a fitting function from Gao et al. [2008]:

$$
\log_{10} c = A \log_{10} M_{\text{halo}} + B.
$$

(13.59)

The parameters are a function of expansion factor, $a$. We use the following fits to the Gao et al. [2008] results:

$$
A = -0.140 \exp \left[ - (\log_{10} a + 0.05)^2 / 0.35 \right],
$$

(13.60)

$$
B = 2.646 \exp \left[ - (\log_{10} a/0.50)^2 \right].
$$

(13.61)

Zhao (2009)

Selected with `darkMatterProfileConcentrationMethod=zhao2009`, under this method the concentration is computed using a fitting function from Zhao et al. [2009]:

$$
c = 4 \left( 1 + \left[ \frac{t}{3.75t_{\text{form}}} \right]^{8.4} \right)^{1/8},
$$

(13.62)

where $t$ is the time for the halo and $t_{\text{form}}$ is a formation time defined by Zhao et al. [2009] as the time at which the main branch progenitor of the halo had a mass equal to 0.04 of the current halo mass. This formation time is computed directly from the merger tree branch associated with each halo. If the no branch exists or does not extend to the formation time then the formation time is computed by extrapolating the mass of the earliest resolved main branch progenitor to earlier times using the selected mass accretion history method (see §17.4.1).
13.8. Dark Matter Halos

Muñoz-Cuartas (2011)

Selected with darkMatterProfileConcentrationMethod=munozCuartas2011, under this method the concentration is computed using a fitting function from Muñoz-Cuartas et al. [2011]:

\[
\log_{10} c = a \log_{10} \left( \frac{M_{\text{halo}}}{h^{-1}M_{\odot}} \right) + b.
\] (13.63)

The parameters are a function of redshift, \( z \), given by

\[
a = wz - m,
\] (13.64)

\[
b = \frac{\alpha}{(z + \gamma)} + \frac{\beta}{(z + \gamma)^2},
\] (13.65)

where \( w = 0.029 \), \( m = 0.097 \), \( \alpha = -110.001 \), \( \beta = 2469.720 \), \( \gamma = 16.885 \).

Prada et al. (2011)

Selected with darkMatterProfileConcentrationMethod=prada2011, under this method the concentration is computed using a fitting function from Prada et al. [2011]:

\[
c(M, t) = B_0(x)C(\sigma'),
\] (13.66)

where

\[
\begin{align*}
\sigma'(M, t) &= B_1(x)\sigma(M, t), \\
B_0(x) &= \frac{c_{\min}(x)}{c_{\min}(1.393)}, \\
B_1(x) &= \frac{\sigma^{-1}_{\min}(x)}{\sigma^{-1}_{\min}(1.393)}, \\
c_{\min}(x) &= c_0 + (c_1 - c_0)[\tan^{-1}\{\alpha(x - x_0)/\Pi + 1/2\}], \\
\sigma^{-1}_{\min}(x) &= \sigma_0^{-1} + (\sigma_1^{-1} - \sigma_0^{-1})[\tan^{-1}\{\beta(x - x_1)/\Pi + 1/2\}], \\
C(\sigma') &= A[(\sigma'/b)^{c + 1}] \exp(d/\sigma'^2), \\
x &= (\Omega_M/\Omega_f)^{1/3}a(t),
\end{align*}
\] (13.67-13.73)

with the following parameters (default values taken from Prada et al. [2011] given in []): \( A = [\text{prada2011ConcentrationA}] = 2.881 \), \( b = [\text{prada2011ConcentrationB}] = 1.257 \), \( c = [\text{prada2011ConcentrationC}] = 1.022 \), \( d = [\text{prada2011ConcentrationD}] = 0.060 \), \( c_0 = [\text{prada2011ConcentrationC0}] = 3.081 \), \( c_1 = [\text{prada2011ConcentrationC1}] = 5.033 \), \( x_0 = [\text{prada2011ConcentrationX0}] = 0.424 \), \( x_1 = [\text{prada2011ConcentrationX1}] = 0.526 \), \( \sigma_0^{-1} = [\text{prada2011ConcentrationInverseSigma1}] = 1.047 \), \( \sigma_1^{-1} = [\text{prada2011ConcentrationInverseSigma1}] = 1.646 \), \( \alpha = [\text{prada2011ConcentrationAlpha}] = 6.948 \), and \( \beta = [\text{prada2011ConcentrationBeta}] = 7.386 \).

Dutton & Macciò (2014)

Selected with darkMatterProfileConcentrationMethod=duttonMaccio2014, under this method the concentration is computed using a fitting function from Dutton and Macciò [2014]:

\[
\log_{10} c = A + B \log_{10} M_{\text{halo}}.
\] (13.74)

The parameters are a function of redshift, \( z \). We use the following fit suggested by Dutton and Macciò [2014] results:

\[
A = A_1 + (A_2 - A_1) \exp[A_3 z^{A_4}]
\]

\[
B = B_1 + B_2 z.
\] (13.75)

The coefficients are chosen from one of the three sets given by Dutton and Macciò [2014], controlled via the [duttonMaccio2014FitType] parameter, as described in Table 13.3.
Table 13.3.: Coefficients appearing in the dark matter halo profile concentration fitting functions of Dutton and Macciò [2014]. The “fit type” is specified by the [duttonMaccio2014FitType] parameter.

<table>
<thead>
<tr>
<th>Fit type</th>
<th>Profile</th>
<th>$\Delta_{\text{vir}}$</th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
<th>$A_4$</th>
<th>$B_1$</th>
<th>$B_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>nfwVirial</td>
<td>NFW Top-hat</td>
<td>+0.537</td>
<td>+1.025</td>
<td>−0.718</td>
<td>+1.080</td>
<td>−0.097</td>
<td>+0.024</td>
<td></td>
</tr>
<tr>
<td>nfw200</td>
<td>NFW</td>
<td>200</td>
<td>+0.520</td>
<td>+0.905</td>
<td>−0.617</td>
<td>+1.210</td>
<td>−0.101</td>
<td>+0.026</td>
</tr>
<tr>
<td>einasto200</td>
<td>Einasto</td>
<td>200</td>
<td>+0.459</td>
<td>+0.977</td>
<td>−0.490</td>
<td>+1.303</td>
<td>−0.130</td>
<td>+0.029</td>
</tr>
</tbody>
</table>

Diemer & Kravtsov (2014)

Selected with [darkMatterProfileConcentrationMethod=diemerKravtsov2014], under this method the concentration is computed using a fitting function from Diemer and Kravtsov [2014]:

$$c = \frac{c_{\text{min}}}{2} \left[ \left( \frac{\nu}{\nu_{\text{min}}} \right)^{-\alpha} + \left( \frac{\nu}{\nu_{\text{min}}} \right)^{\beta} \right],$$

(13.76)

where $c_{\text{min}} = \phi_0 + \phi_1 n$, $\nu_{\text{min}} = \eta_0 + \eta_1 n$, $n$ is the logarithmic slope of the linear power spectrum at wavenumber $k = \kappa 2\pi / R$, $R$ is the comoving Lagrangian radius of the halo, $R = [3M/4\pi \rho_M(z = 0)]^{1/3}$, and $\nu = \delta_{\text{crit}}(t)/\sigma(M)$ is the peak height parameter. The numerical parameters $(\kappa, \phi_0, \phi_1, \eta_0, \eta_1, \alpha, \beta)$ are set by the parameters [darkMatterProfileConcentrationDiemerKravtsov2014Kappa], [darkMatterProfileConcentrationDiemerKravtsov2014Phi0], [darkMatterProfileConcentrationDiemerKravtsov2014Phi1], [darkMatterProfileConcentrationDiemerKravtsov2014Eta0], [darkMatterProfileConcentrationDiemerKravtsov2014Eta1], [darkMatterProfileConcentrationDiemerKravtsov2014Alpha], [darkMatterProfileConcentrationDiemerKravtsov2014Beta], respectively, and default to the values given in Table 3 of Diemer and Kravtsov [2014] for the median relation, namely (0.69, 6.58, 1.37, 6.82, 1.42, 1.12, 1.69).

WDM

Selected with [darkMatterProfileConcentrationMethod=WDM], under this method the concentration is computed by applying the correction factor of Schneider et al. [2012]:

$$c_{\text{WDM}} = c_{\text{CDM}} \left[ 1 + \gamma_1 \left( \frac{M_{1/2}}{M_{\text{halo}}} \right)^{\gamma_2} \right]^{-1},$$

(13.77)

where $\gamma_1 = 15$, $\gamma_2 = 0.3$, $M_{1/2}$ is the mass corresponding to the wavenumber at which the WDM transfer function is suppressed below the CDM transfer function by a factor of 2, and $M_{\text{halo}}$ is the mass of the dark matter halo, to a CDM concentration algorithm as specified by [darkMatterProfileConcentrationCDMMethod].

13.8.4. Density Profile Shape

The method used to compute any shape parameter of dark matter profiles is selected via the [darkMatterShapeMethod] parameter.

Gao (2008)

Selected with [darkMatterShapeMethod=Gao2008], under this method the shape parameter for Einasto density profiles is computed using a fitting function from Gao et al. [2008]:

$$\alpha = \begin{cases} 0.155 + 0.0095\nu^2 & \text{if } \nu < 3.907 \\ 0.3 & \text{if } \nu \geq 3.907, \end{cases}$$

(13.78)
where \( \nu = \delta_c(t) / \sigma(M) \) is the peak height of the halo. The truncation at \( \alpha = 0.3 \) is included since Gao et al. [2008]'s fits do not probe this region and extremely large values of \( \alpha \) are numerically troublesome.

### 13.8.5. Mass Loss Rates

The method used to compute the rate of mass loss from dark matter (sub)halos is selected via the `darkMatterHaloMassLossRateMethod` parameter.

**Null**

Selected with `darkMatterHaloMassLossRateMethod=Null`, this method assumes a zero rate of mass loss from dark matter halos.

**van den Bosch et al. (2005)**

Selected with `darkMatterHaloMassLossRateMethod=vanDenBosch2005`, this method uses the algorithm of van den Bosch et al. [2005] to compute the rate of mass loss. Specifically:

\[
\dot{M}_{\text{node,bound}} = -\frac{M_{\text{node,bound}}}{\tau} (M_{\text{node,bound}}/M_{\text{node,parent}})^\zeta, \tag{13.79}
\]

where \( M_{\text{node,parent}} \) is the mass of the parent node in which the halo lives and

\[
\tau = \tau_0 \left( \frac{\Delta_{\text{vir}}(t)}{\Delta(t_0)} \right)^{-1/2} a^{3/2}, \tag{13.80}
\]

where \( \Delta_{\text{vir}}(t) \) is the virial overdensity of halos at time \( t \) and \( a \) is the expansion factor. The fitting parameters, \( \tau_0 \) and \( \zeta \) have values of 0.13 Gyr and 0.36 respectively as determined by van den Bosch et al. [2005]. Note that van den Bosch et al. [2005] write this expression in a slightly different form since their \( \Delta_{\text{vir}} \) is defined relative to the critical density rather than the mean density as it is in Galacticus. In both cases, the timescale \( \tau \) simply scales as \( \langle \rho_{\text{vir}} \rangle^{-1/2} \) where \( \langle \rho_{\text{vir}} \rangle \) is the mean virial overdensity of halos.

### 13.8.6. Mass Sampling Density Function

The method used to compute the halo mass density function is selected via the `haloMassFunctionSamplingMethod` parameter.

**Power Law**

Selected with `haloMassFunctionSamplingMethod=powerLaw`, under this method the sampling density function is given by

\[
\gamma(M) = \log_{10}(M/M_{\text{minimum}})^{-\alpha/(1+\alpha)}, \tag{13.81}
\]

The resulting distribution of halo masses is such that the mass of the \( i^{th} \) halo is

\[
M_{\text{halo,}i} = \exp \left[ \ln(M_{\text{halo,min}}) + \ln \left( M_{\text{halo,max}}/M_{\text{halo,min}} \right) x_i^{1+\alpha} \right]. \tag{13.82}
\]

Here, \( x_i \) is a number between 0 and 1 and \( \alpha \) = `mergerTreeBuildTreesHaloMassExponent` is an input parameter that controls the relative number of low and high mass tree produced.
13. **Physical Implementations**

### Halo Mass Function

Selected with `haloMassFunctionSamplingMethod=haloMassFunction`, this method sets the sampling density function equal to the halo mass function, \( \gamma(M) = \text{minmax}(\phi_{\text{min}}, \phi_{\text{max}}, \frac{\text{dn}(M)}{d \log M}) \), where \( \phi_{\text{min}} = \text{[haloMassFunctionSamplingAbundanceMinimum]} \), \( \phi_{\text{max}} = \text{[haloMassFunctionSamplingAbundanceMaximum]} \), and

\[
\text{minmax}(a, b, x) = \begin{cases} 
  a & \text{if } x < a \\
  x & \text{if } a \leq x \leq b \\
  b & \text{if } x > b,
\end{cases}
\]

resulting in a sample of halos representative of a volume of space.

### Stellar Mass Function

Selected with `haloMassFunctionSamplingMethod=stellarMassFunction`, in this case the sampling density is chosen to give optimally minimal errors on the model stellar mass function (see §10.4.1 for full details). This calculation requires the observational errors on the stellar mass function to be known. A simple model of the form

\[
\sigma^2(M) = \left[ \phi_0 \left( \frac{M}{M_*} \right)^\alpha \exp \left( - \left[ \frac{M}{M_*} \right]^\beta \right) \right]^2 + \sigma_0^2
\]

is used to model the error, \( \sigma(M) \), on the observed stellar mass function as a function of stellar mass, \( M \), where \( \phi_0 = \text{[haloMassFunctionSamplingStellarMassFunctionErrorPhi0]} \), \( M_* = \text{[haloMassFunctionSamplingStellarMassFunctionErrorMstar]} \), \( \alpha = \text{[haloMassFunctionSamplingStellarMassFunctionErrorAlpha]} \), \( \beta = \text{[haloMassFunctionSamplingStellarMassFunctionErrorBeta]} \), and \( \sigma_0 = \text{[haloMassFunctionSamplingStellarMassFunctionErrorConstant]} \).

### 13.8.7. Spin Parameter Distribution

The method used to compute the distribution of dark matter halo spin parameters is selected via the `haloSpinDistributionMethod` parameter.

#### Lognormal

Selected with `haloSpinDistributionMethod=lognormal`, under this method the spin is drawn from a lognormal distribution with median \([lognormalSpinDistributionMedian]\) and width \([lognormalSpinDistributionSigma]\).

#### Bett et al. (2007)

Selected with `haloSpinDistributionMethod=Bett2007`, under this method the spin is drawn from the distribution found by Bett et al. [2007]. The \( \lambda_0 \) and \( \alpha \) parameter of Bett et al.'s distribution are set by the \([spinDistributionBett2007Lambda0]\) and \([spinDistributionBett2007Alpha]\) input parameters.

#### Delta Function

Selected with `haloSpinDistributionMethod=deltaFunction`, under this method the spin is drawn from a delta function distribution, \( P(\lambda) = \delta(\lambda - \lambda_0) \), where \( \lambda_0 = \text{[deltaFunctionSpinDistributionSpin]} \), i.e. a fixed value of spin equal to \( \lambda_0 \) is returned.

### 13.9. Disk Stability/Bar Formation

The method uses to compute the bar instability timescale for galactic disks is selected via the `barInstabilityMethod` parameter.
13.9.1. Null

Selected with `barInstabilityMethod=null`, this method assumes no instability and so returns an infinite timescale, and no external driving torque.

13.9.2. Efstathiou, Lake & Negroponte

Selected with `barInstabilityMethod=ELN`, this method uses the stability criterion of Efstathiou et al. [1982] to estimate when disks are unstable to bar formation:

\[ \epsilon \left( \equiv \frac{V_{\text{peak}}}{\sqrt{GM_{\text{disk}}/r_{\text{disk}}}} \right) < \epsilon_c, \]  

(13.85)

for stability, where \( V_{\text{peak}} \) is the peak velocity in the rotation curve (computed here assuming an isolated exponential disk), \( M_{\text{disk}} \) is the mass of the disk and \( r_{\text{disk}} \) is its scale length (assuming an exponential disk). The value of \( \epsilon_c \) is linearly interpolated in the disk gas fraction between values for purely gaseous and stellar disks as specified by `stabilityThresholdStellar` and `stabilityThresholdGaseous` respectively. For disks which are judged to be unstable, the timescale for bar formation is estimated to be

\[ t_{\text{bar}} = t_{\text{disk}} \frac{\epsilon_c - \epsilon_{\text{iso}}}{\epsilon_c - \epsilon}, \]

(13.86)

where \( \epsilon_{\text{iso}} \) is the value of \( \epsilon \) for an isolated disk and \( t_{\text{disk}} \) is the disk dynamical time, defined as \( r/V \), at one scale length. This form gives an infinite timescale at the stability threshold, reducing to a dynamical time for highly unstable disks. This method returns zero external driving torque.

13.9.3. Efstathiou, Lake & Negroponte + Tidal Forces

Selected with `barInstabilityMethod=ELN+tidal`, this method uses the stability criterion of Efstathiou et al. [1982] to estimate when disks are unstable to bar formation, but includes an additional term due to external tidal forces:

\[ \epsilon \left( \equiv \frac{V_{\text{peak}}}{\sqrt{GM_{\text{disk}}/r_{\text{disk}}} + \max(T, 0)} \right) < \epsilon_c, \]  

(13.87)

for stability, where \( V_{\text{peak}} \) is the peak velocity in the rotation curve (computed here assuming an isolated exponential disk), \( M_{\text{disk}} \) is the mass of the disk, \( r_{\text{disk}} \) is its scale length (assuming an exponential disk), \( T = FR^2_{\text{disk}} \) is the external driving specific torque, and \( F \) is the external tidal field (evaluated using the selected method; see §13.35). The value of \( \epsilon_c \) is linearly interpolated in the disk gas fraction between values for purely gaseous and stellar disks as specified by `stabilityThresholdStellar` and `stabilityThresholdGaseous` respectively. For disks which are judged to be unstable, the timescale for bar formation is estimated to be

\[ t_{\text{bar}} = t_{\text{disk}} \frac{\epsilon_c - \epsilon_{\text{iso}}}{\epsilon_c - \epsilon}, \]

(13.88)

where \( \epsilon_{\text{iso}} \) is the value of \( \epsilon \) for an isolated disk and \( t_{\text{disk}} \) is the disk dynamical time, defined as \( r/V \), at one scale length. This form gives an infinite timescale at the stability threshold, reducing to a dynamical time for highly unstable disks.

13.10. Excursion Set Barrier

The functional form of the excursion set barrier is selected via the `excursionSetBarrierMethod` parameter.
13. Physical Implementations

13.10.1. Linear Barrier
Selected with `excursionSetBarrierMethod=linear` this method assumes a barrier:

\[ B(S) = B_0 + B_1 S, \]  
(13.89)

where \( B_0 = \text{[excursionSetBarrierConstantCoefficient]} \), and \( B_0 = \text{[excursionSetBarrierLinearCoefficient]} \).

13.10.2. Quadratic Barrier
Selected with `excursionSetBarrierMethod=quadratic` this method assumes a barrier:

\[ B(S) = B_0 + B_1 S + B_2 S^2, \]  
(13.90)

where \( B_0 = \text{[excursionSetBarrierConstantCoefficient]} \), \( B_0 = \text{[excursionSetBarrierLinearCoefficient]} \), and \( B_2 = \text{[excursionSetBarrierQuadraticCoefficient]} \).

13.10.3. Critical Overdensity Barrier
Selected with `excursionSetBarrierMethod=criticalOverdensity` this method assumes a barrier equal to the critical linear theory overdensity for halo collapse (see §13.4.7).

13.11. Excursion Set Barrier First Crossing Distribution
The algorithm to be used when solving the excursion set barrier first crossing distribution problem is selected via the `excursionSetFirstCrossingMethod` parameter.

13.11.1. Linear Barrier
Selected with `excursionSetFirstCrossingMethod=linearBarrier` this method assumes the solution for a linear barrier. Specifically, the first crossing distribution is

\[ f(S,t) = B(0,t) \exp(-B(S,t)^2/2S)/\sqrt{2\pi S}, \]  
(13.91)

where \( B(S,t) \) is the (assumed-to-be-linear-in-\( S \)) barrier at time \( t \) and variance \( S \). The first crossing rate is computed using a finite difference approximation between two closely-spaced times. The non-crossing rate is zero.

13.11.2. Farahi
Selected with `excursionSetFirstCrossingMethod=Farahi` this method solves the barrier first crossing problem using the method of Farahi (see Benson et al. 2012), which proceeds by finding the solution to the integral equation:

\[ 1 = \int_0^S f(S')dS' + \int_{-\infty}^{B(S)} P(\delta,S)d\delta, \]  
(13.92)

where \( P(\delta,S)d\delta \) is the probability for a trajectory to lie between \( \delta \) and \( \delta + d\delta \) at variance \( S \). In the absence of a barrier, \( P(\delta,S) \) would be equal to \( P_0(\delta,S) \) which is simply a Gaussian distribution with variance \( S \):

\[ P_0(\delta,S) = \frac{1}{\sqrt{2\pi S}} \exp\left(-\frac{\delta^2}{2S}\right). \]  
(13.93)
Since the barrier absorbs any random walks which cross it at smaller $S$, the actual $P(\delta, S)$ must therefore be given by:

$$P(\delta, S) = P_0(\delta, S) - \int_0^S f(S')P_0[\delta - B(S'), S - S'] dS'.$$

(13.94)

In the second term on the right hand side of eqn. (13.94) represents the $P_0[\delta - B(S'), S - S']$ term represents the distribution of random trajectories originating from the point $(S, B(S))$. The integral therefore gives the fraction of trajectories which crossed the barrier at $S < S'$ and which can now be found at $(S, \delta)$.

Using this result, we can rewrite eqn. (13.92):

$$1 = \int_0^S f(S')dS' + \int_{-\infty}^{B(S)} \left[ P_0(\delta, S) - \int_0^S f(S')P_0[\delta - B(S'), S - S']dS' \right] d\delta,$$

(13.95)

in general and, for the Gaussian distribution of eqn. (13.93):

$$1 = \int_0^S f(S')dS' + \int_{-\infty}^{B(S)} \left[ \frac{1}{\sqrt{2\pi S}} \exp \left( -\frac{\delta^2}{2S} \right) - \int_0^S f(S') \frac{1}{\sqrt{2\pi (S - S')}} \exp \left( -\frac{[\delta - B(S')]^2}{2(S - S')} \right) dS' \right] d\delta.$$

(13.96)

The integral over $d\delta$ can be carried out analytically to give:

$$1 = \int_0^S f(S')dS' + \text{erf} \left[ \frac{B(S)}{\sqrt{2S}} \right] - \int_0^S f(S') \text{erf} \left[ \frac{B(S) - B(S')}{\sqrt{2(S - S')}} \right] dS''.$$

(13.97)

We now discretize eqn. (13.97). Specifically, we divide the $S$ space into $N$ intervals defined by the points:

$$S_i = \begin{cases} 0 & \text{if } i = 0 \\ \sum_{j=0}^{i-1} \Delta S_i & \text{if } i > 1. \end{cases}$$

(13.98)

Note that $f(0) = 0$ by definition, so $f(S_0) = 0$ always. We choose $\Delta S_i = S_{\max}/N$ (i.e. uniform spacing in $S$) when computing first crossing distributions, and $\Delta S_i \propto S_i$ (i.e. uniform spacing in $\log(S)$) when computing first crossing rates.

Discretizing the integrals in eqn. (13.97) gives:

$$\int_0^{S_j} f(S')dS' = \sum_{i=0}^{j-1} \frac{f(S_i) + f(S_{i+1})}{2} \Delta S_i,$$

(13.99)

and:

$$\int_0^{S_j} f(S') \text{erf} \left[ \frac{B(S) - B(S')}{\sqrt{2(S - S')}} \right] dS' = \sum_{i=0}^{j-1} \frac{1}{2} \left( f(S_i) \text{erf} \left[ \frac{B(S_i) - B(S_j)}{\sqrt{2(S_j - S_i)}} \right] + f(S_{i+1}) \text{erf} \left[ \frac{B(S_j) - B(S_{i+1})}{\sqrt{2(S_j - S_{i+1})}} \right] \right) \Delta S_i.$$

(13.100)

We can now rewrite eqn. (13.97) in discretized form:

$$1 = \sum_{i=0}^{j-1} \frac{f(S_i) + f(S_{i+1})}{2} \Delta S_i + \text{erf} \left[ \frac{B(S_j)}{\sqrt{2S_j}} \right] - \frac{1}{2} \sum_{i=0}^{j-1} \left( f(S_i) \text{erf} \left[ \frac{B(S_i) - B(S_j)}{\sqrt{2(S_j - S_i)}} \right] + f(S_{i+1}) \text{erf} \left[ \frac{B(S_j) - B(S_{i+1})}{\sqrt{2(S_j - S_{i+1})}} \right] \right) \Delta S_i.$$

(13.101)
Solving eqn. (13.101) for $f(S_j)$:
\[
\left(\frac{1}{2} - \frac{1}{2} \text{erf} \left[ \frac{B(S_j) - B(S_j)}{\sqrt{2(S_j - S_j)}} \right] \right) \Delta S_{j-1} f(S_j) = 1 - \sum_{i=0}^{j-2} f(S_i) \frac{f(S_{i+1})}{2} \Delta S_i - \frac{f(S_{j-1})}{2} \Delta S_{j-1} - \text{erf} \left[ \frac{B(S_j)}{\sqrt{2S_j}} \right] \\
+ \frac{1}{2} \sum_{i=0}^{j-2} \left( f(S_i) \text{erf} \left[ \frac{B(S_j) - B(S_i)}{\sqrt{2(S_j - S_i)}} \right] + f(S_{i+1}) \text{erf} \left[ \frac{B(S_j) - B(S_{i+1})}{\sqrt{2(S_j - S_{i+1})}} \right] \right) \Delta S_i \\
+ \frac{1}{2} f(S_{j-1}) \text{erf} \left[ \frac{B(S_j) - B(S_{j-1})}{\sqrt{2(S_j - S_{j-1})}} \right] \Delta S_{j-1}.
\] (13.103)

For all barriers that we consider:
\[
\text{erf} \left[ \frac{B(S_j) - B(S_j)}{\sqrt{2(S_j - S_j)}} \right] = 0.
\] (13.104)

We can then simplify eqn. (13.102):
\[
f(S_j) = \frac{2}{\Delta S_{j-1}} \left[ 1 - \sum_{i=0}^{j-2} f(S_i) \frac{f(S_{i+1})}{2} \Delta S_i - \frac{f(S_{j-1})}{2} \Delta S_{j-1} - \text{erf} \left[ \frac{B(S_j)}{\sqrt{2S_j}} \right] \\
+ \frac{1}{2} \sum_{i=0}^{j-2} \left( f(S_i) \text{erf} \left[ \frac{B(S_j) - B(S_i)}{\sqrt{2(S_j - S_i)}} \right] + f(S_{i+1}) \text{erf} \left[ \frac{B(S_j) - B(S_{i+1})}{\sqrt{2(S_j - S_{i+1})}} \right] \right) \Delta S_i \\
+ \frac{1}{2} f(S_{j-1}) \text{erf} \left[ \frac{B(S_j) - B(S_{j-1})}{\sqrt{2(S_j - S_{j-1})}} \right] \Delta S_{j-1} \right].
\] (13.105)

Consolidating terms in the summations:
\[
f(S_j) = \frac{2}{\Delta S_{j-1}} \left[ 1 - \text{erf} \left[ \frac{B(S_j)}{\sqrt{2S_j}} \right] - \sum_{i=0}^{j-1} \left( 1 - \text{erf} \left[ \frac{B(S_j) - B(S_i)}{\sqrt{2(S_j - S_i)}} \right] \right) f(S_i) \frac{\Delta S_{i+1} + \Delta S_i}{2} \right].
\] (13.106)

In the case of constant $\Delta S_i (= \Delta S)$ this can be simplified further:
\[
f(S_j) = \frac{2}{\Delta S} \left[ 1 - \text{erf} \left[ \frac{B(S_j)}{\sqrt{2S_j}} \right] \right] - \sum_{i=0}^{j-1} \left( 1 - \text{erf} \left[ \frac{B(S_j) - B(S_i)}{\sqrt{2(S_j - S_i)}} \right] \right) f(S_i).
\] (13.107)

In either case (i.e. eqns. 13.105 and 13.106) solution proceeds recursively: $f(S_0) = 0$ by definition, $f(S_1)$ depends only on the known barrier and $f(S_0)$, $f(S_j)$ depends only on the known barrier and $f(S_{<j})$.

The first crossing rate is computed using the same method but with an effective barrier which is offset by the position of the progenitor in the ($\delta, S$) plane, plus a small shift in time. The non-crossing rate is computed directly by integrating over the first crossing rate distribution.


Selected with excursionSetFirstCrossingMethod=ZhangHui2006 this method solves the barrier first crossing problem using the method of Zhang and Hui [2006]. First crossing (and non-crossing) rates are not supported by this method.

Selected with excursionSetFirstCrossingMethod=ZhangHui2006HighOrder this method solves the barrier first crossing problem using a higher-order extension of the method of Zhang and Hui [2006]. First crossing (and non-crossing) rates are not supported by this method.

In this method we discretize the first-crossing distribution function, and use a closed Newton-Cotes method to perform integrations. First, we mesh the $S$ space using uniform spacing in $S$:

$$S_i = i \times \Delta S, i = 0, 1, \ldots, N, \Delta S = \frac{S}{N}. \quad (13.107)$$

Then we discretize the integral equation by Boole’s rule. The integral equation becomes a set of linear algebraic equations:

$$f(S_i) = g_1(S_i) + \frac{\Delta S}{90} \sum_{j=0}^{i-4} \{7f(S_j)g_2(S_i, S_j) + 32f(S_{j+1})g_2(S_i, S_{j+1}) + 12f(S_{j+2})g_2(S_i, S_{j+2}) + 32f(S_{j+3})g_2(S_i, S_{j+3}) + 7f(S_{j+4})g_2(S_i, S_{j+4})\} \quad (13.108)$$

Since $g_2(S, S')$ approaches infinity when $S$ approaches $S'$, one needs to define $g_2(S_i, S_i)$ carefully when $j = i$. We can rewrite the equation:

$$f(S_i) = g_1(S_i) + \frac{4\Delta S}{90} \sum_{j=0}^{i-8} \{7f(S_j)g_2(S_i, S_j) + 32f(S_{j+1})g_2(S_i, S_{j+1}) + 12f(S_{j+2})g_2(S_i, S_{j+2}) + 32f(S_{j+3})g_2(S_i, S_{j+3}) + 7f(S_{j+4})g_2(S_i, S_{j+4})\} + \frac{4\tilde{g}_2(S_i)\Delta S}{90} \left(7f(S_{i-4}) + 32f(S_{i-3}) + 12f(S_{i-2}) + 32f(S_{i-1}) + 7f(S_i)\right) \quad (13.109)$$

For $\tilde{g}_2(S_i)$ we have:

$$\tilde{g}_2(S_i) = \frac{1}{\delta S} \int_{S-\delta S}^{S} g_2(S, S')dS'. \quad (13.110)$$

In the above, $\delta S$ depends on the range of the previous integral part. Generally, $\delta S$ is equal to $4\Delta S$. The above equation can be solved for $f(S_i)$, giving:

$$f(S_i) = \left( g_1(S_i) + \frac{4\Delta S}{90} \sum_{j=0}^{i-8} \{7f(S_j)g_2(S_i, S_j) + 32f(S_{j+1})g_2(S_i, S_{j+1}) + 12f(S_{j+2})g_2(S_i, S_{j+2}) + 32f(S_{j+3})g_2(S_i, S_{j+3}) + 7f(S_{j+4})g_2(S_i, S_{j+4})\} + \frac{4\tilde{g}_2(S_i)\Delta S}{90} \left(7f(S_{i-4}) + 32f(S_{i-3}) + 12f(S_{i-2}) + 32f(S_{i-1})\right) \right) \times \left(1 - \frac{4\tilde{g}_2(S_i)\Delta S}{90}\right)^{-1}. \quad (13.111)$$

Not all of the $i$'s are divisible by 4. So, for the first $m^{th}$ spaces, we need to calculate the integral part, separately, where $m$ is the remainder of $i$ by the modulus of 4. It is a good approximation to calculate
the first part linearly. Consequently, the final formula for the general problem is:

\[
f(S_i) = \left( g_1(S_i) + \frac{\Delta S}{2} \sum_{j=0}^{m-1} \left( f(S_j)g_2(S_i, S_j) + f(S_{j+1})g_2(S_i, S_{j+1}) \right) \right. \\
+ \frac{4\Delta S}{90} \sum_{i-8}^{i-1} \{7f(S_j)g_2(S_i, S_j) + 32f(S_{j+1})g_2(S_i, S_{j+1}) \\
+ 12f(S_{j+2})g_2(S_i, S_{j+2}) + 32f(S_{j+3})g_2(S_i, S_{j+3}) + 7f(S_{j+4})g_2(S_i, S_{j+4}) \} \\
+ \frac{4\tilde{g}_2(S_i)\Delta S}{90} \left( 7f(S_{i-4}) + 32f(S_{i-3}) + 12f(S_{i-2}) + 32f(S_{i-1}) \right) \\
\left. \times \left( 1 - \frac{4\tilde{g}_2(S_i)\Delta S}{90} \right)^{-1} \right) \\
\tag{13.112}
\]

Finally, we can solve the first-crossing distribution function, giving:

\[
f(S_0) = g_1(S_0) = 0 \\
f(S_1) = g_1(S_1) \left( 1 - \frac{\tilde{g}_2(S_1)\Delta S}{2} \right)^{-1} \\
f(S_2) = \left( g_1(S_2) + \frac{\Delta S}{2}2\tilde{g}_2(S_2)f(S_1) \right) \left( 1 - \frac{\tilde{g}_2(S_2)\Delta S}{2} \right)^{-1} \\
f(S_3) = \left( g_1(S_3) + \frac{\Delta S}{2}2\tilde{g}_2(S_3)(f(S_1) + f(S_2)) \right) \left( 1 - \frac{\tilde{g}_2(S_3)\Delta S}{2} \right)^{-1} \\
f(S_i) = \left( g_1(S_i) + \frac{\Delta S}{2} \sum_{j=0}^{m-1} \left( f(S_j)\tilde{g}_2(S_i, S_j) + f(S_{j+1})\tilde{g}_2(S_i, S_{j+1}) \right) \\
+ \frac{4\Delta S}{90} \sum_{i-8}^{i-1} \{7f(S_j)\tilde{g}_2(S_i, S_j) + 32f(S_{j+1})\tilde{g}_2(S_i, S_{j+1}) \\
+ 12f(S_{j+2})\tilde{g}_2(S_i, S_{j+2}) + 32f(S_{j+3})\tilde{g}_2(S_i, S_{j+3}) + 7f(S_{j+4})\tilde{g}_2(S_i, S_{j+4}) \} \\
+ \frac{4\tilde{g}_2(S_i)\Delta S}{90} \left( 7f(S_{i-4}) + 32f(S_{i-3}) + 12f(S_{i-2}) + 32f(S_{i-1}) \right) \\
\times \left( 1 - \frac{4\tilde{g}_2(S_i)\Delta S}{90} \right)^{-1}, \\
\tag{13.117}
\]

where \( \tilde{g}_2(S_1) \) is defined as:

\[
\tilde{g}_2(S_1) = \frac{1}{\Delta S} \int_0^{S_1} g_2(S, S')dS' \text{ if } i = 1 \\
\tilde{g}_2(S_2) = \frac{1}{2\Delta S} \int_0^{S_2} g_2(S, S')dS' \text{ if } i = 0 \\
\tilde{g}_2(S_3) = \frac{1}{3\Delta S} \int_0^{S_3} g_2(S, S')dS' \text{ if } i = 0 \\
\tilde{g}_2(S_i) = \frac{1}{4\Delta S} \int_{S_{i-4}\Delta S}^{S_i} g_2(S, S')dS' \text{ if } i > 3. \\
\tag{13.118-13.121}
\]
The error term for this method of discretization is:

\[ \epsilon = \frac{(\delta S)^7}{1935360} f^{(6)}(\xi), \]  

(13.122)

where \( f^{(6)}(\xi) \) is the absolute maximum of the sixth derivative in the range of \([S_i, S_i + \delta S]\). For this problem, \( \delta S = 4\Delta S \), so:

\[ \epsilon \leq \frac{(\Delta S)^7}{118.125} f^{(6)}(\xi). \]  

(13.123)

Since there are \( N/4 \) intervals, the maximum deviation from the real value of the function is:

\[ \epsilon \leq \sum_{i=1}^{N} \frac{N}{4} \frac{(\Delta S)^7}{118.125} f^{(6)}(\xi_i) \leq N\frac{4}{472.5} (\Delta S)^7 f^{(6)}(\xi), \]

where \( f^{(6)}(\xi) \) is the absolute maximum of the sixth derivative in the domain, \([0, S]\).

### 13.12. Excursion Set Barrier Remapping

Remappings of the excursion set barrier are selected via the `excursionSetBarrierRemapMethods` (for calculations of first crossing distributions) and `excursionSetBarrierRatesRemapMethods` (for calculations of first crossing rate distributions) parameters. The parameters may specify multiple remappings—these will be applied to the barrier in order.

#### 13.12.1. Null Remapping

Selected with `excursionSetBarrier(Rates)RemapMethods=null` this method leaves the barrier unmodified.

#### 13.12.2. Scale Remapping

Selected with `excursionSetBarrier(Rates)RemapMethods=scale` this method multiplies the barrier by [`excursionSetBarrierRemapScalingFactor`].

#### 13.12.3. Sheth-Mo-Tormen Remapping

Selected with `excursionSetBarrier(Rates)RemapMethods=Sheth-Mo-Tormen` this method remaps the barrier according to the algorithm of Sheth et al. [2001]:

\[ B(S) \rightarrow \sqrt{A}B(S) \left( 1 + b \left[ \frac{S}{AB^2(S)} \right]^c \right), \]  

(13.124)

where \( A = 0.707 \), \( b = 0.5 \), and \( c = 0.6 \).

### 13.13. Galactic Structure

The algorithm to be used when solving for galactic structure (specifically, finding radii of galactic components) is selected via the `galacticStructureRadiusSolverMethod` parameter.
13. Physical Implementations

13.13.1. Fixed

Selected with `galacticStructureRadiusSolverMethod=fixed` this method determines the sizes of galactic components by assuming that radius equals

\[ r = f_r \lambda r_{\text{vir}} \]  \hspace{1cm} (13.125)

where \( r_{\text{vir}} \) is the virial radius of the node, \( \lambda \) is its spin parameter and \( f_r = [\text{galacticStructureRadiiFixedFactor}] \) is a parameter.

13.13.2. Linear

Selected with `galacticStructureRadiusSolverMethod=linear` this method determines the sizes of galactic components by assuming that radius scales linearly with specific angular momentum such that

\[ r = r_{\text{vir}} j / j_{\text{vir}} \]  \hspace{1cm} (13.126)

where \( j \) is the specific angular momentum of the component (at whatever point in the profile is to be solved for), \( r \) is radius, \( r_{\text{vir}} \) is the virial radius of the node and \( j_{\text{vir}} = r_{\text{vir}} v_{\text{vir}} \) with \( v_{\text{vir}} \) being the virial velocity of the node.

13.13.3. Simple

Selected with `galacticStructureRadiusSolverMethod=simple` this method determines the sizes of galactic components by assuming that their self-gravity is negligible (i.e. that the gravitational potential well is dominated by dark matter) and that, therefore, baryons do not modify the dark matter density profile. The radius of a given component is then found by solving

\[ j = \sqrt{G M_{\text{DM}}(r) r} \]  \hspace{1cm} (13.127)

where \( j \) is the specific angular momentum of the component (at whatever point in the profile is to be solved for), \( r \) is radius and \( M(r) \) is the mass of dark matter within radius \( r \). The parameter \([\text{adiabaticContractionUseFormationHalo}]\) controls whether the structure of the galaxy will be solved for using the properties of its present node or those of its node at the time of node formation (which requires that “node formation” has been suitably defined and implemented by a component).

13.13.4. Adiabatic

Selected with `galacticStructureRadiusSolverMethod=adiabatic`, this method takes into account the baryonic self-gravity of all galactic components when solving for structure and additionally accounts for backreaction of the baryons on the dark matter density profile. Solution proceeds via an iterative procedure to find equilibrium radii for all galaxies in a consistently contracted halo. To account for adiabatic contraction the mass of dark matter within a given radius \( r_f \) is taken to be equal to that originally within radius \( r_i \) in the uncontracted halo. This initial radius is computed using the selected galactic structure solver initial radius algorithm (see §13.14). The parameter \([\text{adiabaticContractionUseFormationHalo}]\) controls whether the structure of the galaxy will be solved for using the properties of its present node or those of its node at the time of node formation (which requires that “node formation” has been suitably defined and implemented by a component).


The algorithm to be used when solving for the initial radius in the dark matter halo when computing galactic structure is selected via the `galacticStructureRadiusSolverInitialRadiusMethod` parameter.

Selected with \texttt{galacticStructureRadiusSolverInitialRadiusMethod=static} this method assumes that the dark matter halo is static and so sets the initial radius equal to the final radius.

13.14.2. Adiabatic

Selected with \texttt{galacticStructureRadiusSolverInitialRadiusMethod=adiabatic} this method assumes that adiabatic contraction follows the algorithm of Gnedin et al. [2004]. The parameters $A$ and $\omega$ of that model are specified via input parameters \texttt{adiabaticContractionGnedinA} and \texttt{adiabaticContractionGnedinOmega} respectively.

Given the final radius, $r_f$, the corresponding initial radius, $r_i$, is found by solving:

\[
 f_i M_{\text{total,}0}(\bar{r}_i) r_i = f_i M_{\text{total,}0}(\bar{r}_i) r_i + V_b^2(\bar{r}_i) \bar{r}_i r_i / G,
\]

(13.128)

where $M_{\text{total,}0}(r)$ is the initial total matter profile, $V_b(r)$ is the baryonic contribution to the rotation curve, $f_i$ is the fraction of mass within the virial radius compared to the node mass\footnote{In \texttt{Galacticus} the “node mass” refers to the total mass of the node, assuming it has the universal complement of baryons. Since some halos may contain less than the complete complement of baryons it is possible that $f_i < 1.$}, $f_i = (\Omega_M - \Omega_b) / \Omega_M + M_{\text{satellite,baryonic}} / M_{\text{total}}$, $M_{\text{satellite,baryonic}}$ is the baryonic mass in any satellite halos, $M_{\text{total}}$ is the node mass, and

\[
 \frac{\bar{r}}{r_{\text{vir}}} = A \left( \frac{r}{r_{\text{vir}}} \right)^{\omega - 1},
\]

(13.129)

where $r_{\text{vir}}$ is the virial radius. Note that we explicitly assume that the initial, uncontracted total density profile has the same shape as the initial dark matter density profile, that contraction of the halo occurs with no shell crossing, and that satellite halos trace the dark matter profile of their host halo.

The derivative, $d\bar{r}/d\bar{r}_f \equiv r_i'$ is found by taking the derivative of eqn. (13.128) to give:

\[
 f_i M_{\text{total,}0}(\bar{r}_i) r_i' + f_i 4\pi \bar{r}_i^2 \rho_{\text{total,}0}(\bar{r}_i) \frac{d\bar{r}_i}{dr} r_i' = f_i M_{\text{total,}0}(\bar{r}_i) + f_i 4\pi \bar{r}_i^2 \rho_{\text{total,}0}(\bar{r}_i) \frac{d\bar{r}_i}{dr} r_i' + V_b^2(\bar{r}_i) \bar{r}_i / G + V_b^2(\bar{r}_i) \frac{d\bar{r}_i}{dr} (\bar{r}_i) \bar{r}_i / G + \bar{r}_i / G,
\]

(13.130)

where

\[
 \frac{d\bar{r}}{dr} = A \left( \frac{r}{r_{\text{vir}}} \right)^{-1},
\]

(13.131)

and which can then be solved numerically for $r_i'$.

13.15. Galaxy Merging

The process of merging two galaxies currently involves two algorithms: one which decides how the merger causes mass components from both galaxies to move and one which determines the size of the remnant galaxy spheroid.

13.15.1. Mass Movements

The movement of mass elements in the merging galaxies is determined by the \texttt{satelliteMergingMassMovementsMethod} parameter.
13. Physical Implementations

**Very Simple**

Selected with `satelliteMergingMassMovementsMethod=verySimple`, this method assumes that the satellite material is always added to the disk of the host, while the host mass is not moved.

**Simple**

Selected with `satelliteMergingMassMovementsMethod=simple`, this method implements mass movements according to:

- If \( M_{\text{satellite}} > f_{\text{major}} M_{\text{central}} \) then all mass from both satellite and central galaxies moves to the spheroid component of the central galaxy;
- Otherwise: Gas from the satellite moves to the component of the central specified by the \( \text{minorMergerGasMovesTo} \) parameter (either “disk” or “spheroid”), stars from the satellite moves to the spheroid of the central and mass in the central does not move.

Here, \( f_{\text{major}} = [\text{majorMergerMassRatio}] \) is the mass ratio above which a merger is considered to be “major”.

**Baugh et al. (2005)**

Selected with `satelliteMergingMassMovementsMethod=Baugh2005`, this method implements mass movements according to:

- If \( M_{\text{satellite}} > f_{\text{major}} M_{\text{central}} \) then all mass from both satellite and central galaxies moves to the spheroid component of the central galaxy;
- Otherwise:
  - If \( M_{\text{central, spheroid}} < f_{\text{burst}} M_{\text{central}} \) and the gas fraction in the host equals or exceeds \( f_{\text{gas, crit}} \) then all gas is moved to the host spheroid, while the host stellar disk remains in place.
  - Otherwise, gas from the satellite moves to the component of the central specified by the \( \text{minorMergerGasMovesTo} \) parameter (either “disk” or “spheroid”), stars from the satellite moves to the spheroid of the central and mass in the central does not move.

Here, \( f_{\text{major}} = [\text{majorMergerMassRatio}] \) is the mass ratio above which a merger is considered to be “major”, while \( f_{\text{burst}} = [\text{burstMassRatio}] \) and \( f_{\text{gas, crit}} = [\text{burstCriticalGasFraction}] \).

13.15.2. Remnant Sizes

The method used to calculate the sizes of merger remnant spheroids is selected by the `satelliteMergingRemnantSizeMethod` parameter.

**Null**

Selected using `satelliteMergingRemnantSizeMethod=null`, this is a null method which does nothing at all. It is useful, for example, when running GALACTICUS to study dark matter only (i.e. when no galaxy properties are computed).
13.15. Galaxy Merging

Cole et al. (2000)
Selected using satelliteMergingRemnantSizeMethod=Cole2000, this method uses the algorithm of Cole et al. [2000] to compute merger remnant spheroid sizes. Specifically

\[
\frac{(M_1 + M_2)^2}{r_{new}} = \frac{M_1^2}{r_1} + \frac{M_2^2}{r_2} + \frac{f_{\text{orbit}} M_1 M_2}{c (r_1 + r_2)},
\]

(13.132)

where \(M_1\) and \(M_2\) are the baryonic masses of the components of the merging galaxies that will end up in the spheroid component of the remnant and \(r_1\) and \(r_2\) are the half mass radii of those same components of the merging galaxies, \(r_{\text{new}}\) is the half mass radius of the spheroidal component of the remnant galaxy and \(c\) is a constant which depends on the distribution of the mass. For a Hernquist spheroid \(c = 0.40\) can be found by numerical integration while for an exponential disk \(c = 0.49\). For simplicity a value of \(c = 0.5\) is adopted for all components. The parameter \(f_{\text{orbit}} = \text{mergerRemnantSizeOrbitalEnergy}\) depends on the orbital parameters of the galaxy pair. For example, a value of \(f_{\text{orbit}} = 1\) corresponds to point mass galaxies in circular orbits about their center of mass.

A subtlety arises because the above expression accounts for only the baryonic mass of material which becomes part of the spheroid component of the remnant. In reality, there are additional terms in the energy equation due to the interaction of this material with any dark matter mass in each galaxy and any baryonic mass of each galaxy which does not become part of the spheroid component of the remnant. To account for this additional matter, an effective boost factor, \(f_{\text{boost}}\), to the specific angular momentum of each component of each merging galaxy is computed:

\[
f_{\text{boost}} = \frac{j}{\sqrt{GM r_{1/2}}},
\]

(13.133)

where \(j\) is the specific angular momentum of the component, \(M\) is its total baryonic mass and \(r_{1/2}\) is its half-mass radius. The mass-weighted mean boost factor is found by combining those of all components which will form part of the spheroid of the remnant. The final specific angular momentum of the remnant spheroid is then given by:

\[
\dot{j}_{\text{new}} = \langle f_{\text{boost}} \rangle r_{\text{new}} V_{\text{new}},
\]

(13.134)

where

\[
V_{\text{new}}^2 = \frac{G(M_1 + M_2)}{r_{\text{new}}}.
\]

(13.135)

Covington et al. (2008)
Selected using satelliteMergingRemnantSizeMethod=Covington2008, this method uses the algorithm of Covington et al. [2008] to compute merger remnant spheroid sizes. Specifically

\[
\frac{(M_1 + M_2)^2}{r_{\text{new}}} = \left[ \frac{M_1^2}{r_1} + \frac{M_2^2}{r_2} + \frac{f_{\text{orbit}} M_1 M_2}{c (r_1 + r_2)} \right] (1 + f_{\text{gas}} C_{\text{rad}}),
\]

(13.136)

where \(M_1\) and \(M_2\) are the baryonic masses of the merging galaxies and \(r_1\) and \(r_2\) are their half mass radii, \(r_{\text{new}}\) is the half mass radius of the spheroidal component of the remnant galaxy and \(c\) is a constant which depends on the distribution of the mass. For a Hernquist spheroid \(c = 0.40\) can be found by numerical integration while for an exponential disk \(c = 0.49\). For simplicity a value of \(c = 0.5\) is adopted for all components. The parameter \(f_{\text{orbit}} = \text{mergerRemnantSizeOrbitalEnergy}\) depends on the orbital

\(^2\)Depending on the merging rules (see §13.15.1) not all mass may be placed into the spheroid component of the remnant.

\(^3\)In practice, GALACTICUS computes a weighted average of the disk and spheroid half-mass radii of each galaxy, with weights equal to the masses of each component (disk and spheroid) which will become part of the spheroid component of the remnant.
parameters of the galaxy pair. For example, a value of \( f_{\text{orbit}} = 1 \) corresponds to point mass galaxies in circular orbits about their center of mass. The final term on the right hand side of eqn. (13.136) gives a correction to the final energy of the remnant due to dissipational losses based on the results of Covington et al. [2011], with

\[
  f_{\text{gas}} = \frac{M_{1,\text{gas}} + M_{2,\text{gas}}}{M_1 + M_2}
\]

begin the gas fraction of the progenitor galaxies. By default, \( C_{\text{rad}} = 2.75 \) [Covington et al., 2011]. To account for the effects of dark matter and non-spheroid baryonic matter the same approach is used as in the Cole et al. [2000] algorithm (see §13.15.2).

13.15.3. Progenitor Properties

The method used to calculate the properties of merger progenitor galaxies is selected by the \texttt{satelliteMergingRemnantProgenitorPropertiesMethod} parameter.

\textbf{Cole et al. (2000)}

Selected using \texttt{satelliteMergingRemnantProgenitorPropertiesMethod=}Cole2000, this method uses the algorithms of Cole et al. [2000] to compute progenitor properties. Masses of progenitors are set to

\[
  M_{\text{host}|\text{satellite}} = \sum_{i=\text{disk}|\text{spheroid}} \sum_{j=\text{stars}|\text{gas}} M_{i,j},
\]

where \( M_{i,j} \) is the mass of mass type \( j \) in component \( i \). Masses of progenitors that will end up in the remnant spheroid are set to

\[
  M_{\text{spheroid host}|\text{satellite}} = \sum_{i=\text{disk}|\text{spheroid}} \sum_{j=\text{stars}|\text{gas}} M_{i,j} \delta_{i,j},
\]

where \( \delta_{i,j} = 0 \) of mass type \( j \) in component \( i \) will end up in the remnant spheroid and 0 otherwise. Radii of material that will end up in the spheroid are set by finding the solution to:

\[
  \sum_{i=\text{disk}|\text{spheroid}} \sum_{j=\text{stars}|\text{gas}} M_{i,j}(r) \delta_{i,j} = \frac{1}{2} \sum_{i=\text{disk}|\text{spheroid}} \sum_{j=\text{stars}|\text{gas}} M_{i,j} \delta_{i,j},
\]

such that the radii are the half-mass radii of the material that will end up in the remnant spheroid. Finally, the angular momentum factor is set to

\[
  f_{\text{AM host}|\text{satellite}} = \frac{1}{M_{\text{spheroid host}|\text{satellite}}} \sum_{i=\text{disk}|\text{spheroid}} \sum_{j=\text{stars}|\text{gas}} M_{i,j} \frac{J_{i,j}}{G M_{i,j}^{3/2} r_{1/2} i,j} \delta_{i,j},
\]

where \( J_{i,j} \) is the angular momentum or pseudo-angular momentum of mass type \( j \) in component \( i \).\footnotemark

\footnotetext{This is technically not quite what Cole et al. [2000] do. Instead, when computing the masses of the material which ends up in the spheroid they include twice the mass of dark matter (accounting for the effects of adiabatic contraction) within the half-mass radius of each galaxy (as calculated above). The final angular momentum is then \( j = \sqrt{G M_{\text{remnant}} r_{1/2}} \) (where \( M_{\text{remnant}} \) includes the contribution from dark matter and the factor of 2 appears to make this the half-mass). This approach is currently not used in \textsc{Galacticus} since there is no way to get the mass of dark matter enclosed accounting for adiabatic contraction in the general case. This is a solvable problem, and so this algorithm is expected to be modified to match that of Cole et al. [2000] precisely in a future version of \textsc{Galacticus}.}
13.16. Hot Halo Mass Distribution

Standard

Selected using satelliteMergingRemnantProgenitorPropertiesMethod=standard, this is the standard method to compute progenitor properties. Masses of progenitors are set to

\[ M_{\text{host|satellite}} = \sum_{i=\text{disk|sph}eroid} \sum_{j=\text{stars|gas}} M_{i,j}, \]  

(13.142)

where \( M_{i,j} \) is the mass of mass type \( j \) in component \( i \). Masses of progenitors that will end up in the remnant spheroid are set to

\[ M_{\text{spheroid host|satellite}} = \sum_{i=\text{disk|sph}eroid} \sum_{j=\text{stars|gas}} M_{i,j} \delta_{i,j}, \]  

(13.143)

where \( \delta_{i,j} = 0 \) of mass type \( j \) in component \( i \) will end up in the remnant spheroid and 0 otherwise. Radii of material that will end up in the spheroid are set to

\[ r_{\text{host|satellite}} = \frac{1}{M_{\text{spheroid host|satellite}}} \sum_{i=\text{disk|sph}eroid} \sum_{j=\text{stars|gas}} M_{i,j} r_{1/2 \text{i,j}} \delta_{i,j}. \]  

(13.144)

Finally, the angular momentum factor is set to

\[ f_{\text{AM host|satellite}} = \frac{1}{M_{\text{spheroid host|satellite}}} \sum_{i=\text{disk|sph}eroid} \sum_{j=\text{stars|gas}} M_{i,j} \frac{J_{i,j}}{G M_{i,j}^{3/2} r_{1/2 \text{i,j}}} \delta_{i,j}, \]  

(13.145)

where \( J_{i,j} \) is the angular momentum or pseudo-angular momentum of mass type \( j \) in component \( i \).

13.16. Hot Halo Mass Distribution

The algorithm to be used when determining the hot halo mass distribution is selected via the hotHaloMassDistributionMethod parameter.

13.16.1. \( \beta \)-profile

Selected with hotHaloMassDistributionMethod=betaProfile this method adopts a spherically symmetric \( \beta \)-profile density profile for the hot halo. Specifically,

\[ \rho_{\text{hot halo}}(r) \propto \left[ r^2 + r_{\text{core}}^2 \right]^{3\beta/2}, \]  

(13.146)

where the core radius, \( r_{\text{core}} \), is set using the selected cored profile core radius method (see §13.17). The value of \( \beta \) is specified by the [hotHaloMassDistributionBeta] parameter. The profile is normalized such that the current mass in the hot gas profile is contained within the outer radius of the hot halo, \( r_{\text{hot,outer}} \).


Selected with hotHaloMassDistributionMethod=ricotti2000 this method adopts a spherically symmetric \( \beta \)-profile density profile for the hot halo with parameters selected using the fitting function of Ricotti and Shull [2000] who found these by solving for hydrostatic equilibrium in an NFW density profile. Specifically, \( \beta = 0.96 \) where

\[ b = \frac{2c}{9} \left[ \log(1+c) - \frac{c}{1+c} \right]^{-1}, \]  

(13.147)
13. Physical Implementations

c is the concentration parameter of the dark matter halo, and $\Gamma$ is the ratio of virial and gas temperatures, which is assumed to be unity. The core radius is $r_{\text{core}} = 0.22r_s$ where $r_s$ is the scale radius of the dark matter profile. The profile is normalized such that the current mass in the hot gas profile is contained within the outer radius of the hot halo, $r_{\text{hot,outer}}$.

13.16.3. Null

Selected with $\text{hotHaloMassDistributionMethod} = \text{null}$ this method assumes no hot halo mass distribution. It is useful, for example, when performing dark matter-only calculations.

13.17. Hot Halo Mass Distribution Profile: Cored Profile Core Radius

The algorithm to be used when determining the core radius of cored isothermal hot halo density profiles is selected via the $\text{hotHaloMassDistributionCoreRadiusMethod}$ parameter.

13.17.1. Growing Core

Selected with $\text{hotHaloMassDistributionCoreRadiusMethod} = \text{growingCore}$, this method implements a core radius equal to a fraction $[\text{coreRadiusOverScaleRadius}]$ of the node’s dark matter profile scale radius for nodes containing a mass of hot gas equal to the universal baryon fraction times their total mass. For nodes containing less hot gas mass, the core radius is expanded to maintain the same gas density at the virial radius, with a maximum core radius of $[\text{coreRadiusOverVirialRadiusMaximum}]$ times the node’s virial radius.

13.17.2. Virial Fraction

Selected with $\text{hotHaloMassDistributionCoreRadiusMethod} = \text{virialRadiusFraction}$, this method implements a core radius equal to a fraction $[\text{coreRadiusOverVirialRadius}]$ of the node’s virial radius.

13.18. Hot Halo Ram Pressure Force

The algorithm to be used when determining the ram pressure force due to the hot halo is selected via the $\text{hotHaloRamPressureForceMethod}$ parameter.

13.18.1. Null

Selected with $\text{hotHaloRamPressureForceMethod} = \text{null}$ this method assumes a zero ram pressure force due to the hot halo.

13.18.2. Font et al. (2008)

Selected with $\text{hotHaloRamPressureForceMethod} = \text{Font2008}$ this method computes the ram pressure force using the algorithm of Font et al. [2008]. Specifically, the ram pressure force is

$$F_{\text{ram,hot,host}} = \rho_{\text{hot,host}}(r_{\text{peri}}) v^2(r_{\text{peri}}),$$

(13.148)

where $\rho_{\text{hot,host}}(r)$ is the hot halo density profile of the node’s host halo, $v(r)$ is the orbital velocity of the node in that host, and $r_{\text{peri}}$ is the pericentric radius of the node’s orbit.
13.19. Hot Halo Ram Pressure Stripping Radius

The algorithm to be used when determining the radius to which the hot halo is stripped by ram pressure forces is selected via the hotHaloRamPressureStrippingMethod parameter.

13.19.1. Virial Radius

Selected with hotHaloRamPressureStrippingMethod= virialRadius this method sets the ram pressure stripping radius equal to the virial radius of the halo. The effectively results in no ram pressure stripping.


Selected with hotHaloRamPressureStrippingMethod= Font2008 this method computes the ram pressure stripping radius using the algorithm of Font et al. [2008]. Specifically, the radius, \( r_p \), is computed as the solution of

\[
\alpha r_p \frac{G M_{\text{satellite}}(r_p) \rho_{\text{hot, satellite}}(r_p)}{r_p} = F_{\text{ram, hot, host}},
\]

where \( M_{\text{satellite}}(r) \) is the total mass of the satellite within radius \( r \), \( F_{\text{ram, hot, host}} \) is the ram pressure force due to the hot halo (computed using the selected hot halo ram pressure force method; see §13.18). The parameter \( \alpha_r \approx [\text{ramPressureStrippingFormFactor}] \) is a geometric factor of order unity.

13.20. Hot Halo Ram Pressure Stripping Timescale

The algorithm to be used when determining the timescale on which the hot halo is stripped by ram pressure forces is selected via the hotHaloRamPressureStrippingTimescaleMethod parameter.

13.20.1. Halo Dynamical Timescale

Selected with hotHaloRamPressureStrippingTimescaleMethod= haloDynamicalTime this method sets the ram pressure stripping timecale equal to the dynamical time of the associated halo.

13.20.2. Ram Pressure Acceleration

Selected with hotHaloRamPressureStrippingTimescaleMethod= ramPressureAcceleration this method computes the ram pressure stripping timescale from the acceleration imparted by the ram pressure force. Following Roediger and BrÃČÅŠggen [2007] this is approximated as:

\[
a_{\text{rampressure}} = \frac{P_{\text{rampressure}}}{\Sigma},
\]

where \( P_{\text{rampressure}} \) is the ram pressure force per unit area, and \( \Sigma \) is the surface density of gas. The associated timescale to accelerate gas over a distance \( r_{\text{outer}} \) (the current outer radius of the hot halo) is then:

\[
\tau_{\text{rampressure}} = \sqrt{\frac{2r_{\text{outer}} \Sigma_{\text{outer}}}{P_{\text{rampressure}}}}.
\]

13.21. Hot Halo Temperature Profile

The algorithm to be used when determining the hot halo temperature profile is selected via the hotHaloTemperatureMethod parameter.
13. Physical Implementations

13.21.1. Virial Temperature

Selected with `hotHaloTemperatureMethod=virial` this method assumes an isothermal halo with a temperature equal to the virial temperature of the halo.

13.22. Initial Mass Functions

The stellar IMF subsystem supports multiple IMFs and extensible algorithms to select which IMF to use based on the physical conditions of star formation.

13.22.1. Initial Mass Function Selection

The method to use for selecting which IMF to use is specified by the `imfSelectionMethod` parameter.

**Fixed**

Selected by `imfSelectionMethod=fixed`, this method uses a fixed IMF irrespective of physical conditions. The IMF to use is specified by the `imfSelectionFixed` parameter (e.g. setting this parameter to `Salpeter` selects the Salpeter IMF).

**Disk and Spheroid**

Selected by `imfSelectionMethod=diskSpheroid`, this method uses different IMFs for star formation in disks and spheroids irrespective of other physical conditions. The IMFs to use are specified by the `imfSelectionDisk` and `imfSelectionSpheroid` parameters (e.g. setting one of these parameters to `Salpeter` selects the Salpeter IMF).

13.22.2. Initial Mass Functions

A variety of different IMFs are available. Each IMF registers itself with the GALACTICUS IMF subsystem and can then be looked-up by name or internal index. All IMFs are assumed to be continuous in $M$, unless otherwise noted and normalized to unit mass. Each IMF supplies a recycled fraction and metal yield for use in the instantaneous recycling approximation. These can be set via the parameters `imf[imfName]RecycledInstantaneous` and `imf[imfName]YieldInstantaneous` where `[imfName]` is the name of the IMF. Their default values were computed using GALACTICUS’s internal stellar astrophysics modules for a Solar metallicity population with age of 13.8 Gyr.

Baugh et al. (2005) Top-Heavy

The Baugh2005TopHeavy IMF is defined by [Baugh et al., 2005]:

$$\phi(M) \propto M^{-1} \text{ for } 0.15M_\odot < M < 125M_\odot$$

(13.152)

Chabrier

The Chabrier IMF is defined by [Chabrier, 2001]:

$$\phi(M) \propto \begin{cases} M^{-1} \exp\left(-[\log_{10}(M/M_c)/\sigma_c]^2/2\right) \text{ for } 0.1M_\odot < M < 1M_\odot \\ M^{-2.3} \text{ for } 1M_\odot < M < 125M_\odot \\ 0 \text{ otherwise,} \end{cases}$$

(13.153)

where $\sigma_c = 0.69$ and $M_c = 0.08M_\odot$. 

278
Kennicutt

The Kennicutt IMF is defined by [Kennicutt, 1983]:

\[
\phi(M) \propto \begin{cases} 
    M^{-1.25} & \text{for } 0.10 M_\odot < M < 1.00 M_\odot \\
    M^{-2.00} & \text{for } 1.00 M_\odot < M < 2.00 M_\odot \\
    M^{-2.30} & \text{for } 2.00 M_\odot < M < 125 M_\odot \\
    0 & \text{otherwise.} 
\end{cases} 
\] (13.154)

Kroupa

The Kroupa IMF is defined by [Kroupa, 2001]:

\[
\phi(M) \propto \begin{cases} 
    M^{-0.3} & \text{for } 0.01 M_\odot < M < 0.08 M_\odot \\
    M^{-1.8} & \text{for } 0.08 M_\odot < M < 0.5 M_\odot \\
    M^{-2.7} & \text{for } 0.5 M_\odot < M < 1 M_\odot \\
    M^{-2.3} & \text{for } 1 M_\odot < M < 125 M_\odot \\
    0 & \text{otherwise.} 
\end{cases} 
\] (13.155)

Miller-Scalo

The Miller-Scalo IMF is defined by [Miller and Scalo, 1979]:

\[
\phi(M) \propto \begin{cases} 
    M^{-1.25} & \text{for } 0.10 M_\odot < M < 1.00 M_\odot \\
    M^{-2.00} & \text{for } 1.00 M_\odot < M < 2.00 M_\odot \\
    M^{-2.30} & \text{for } 2.00 M_\odot < M < 10.0 M_\odot \\
    M^{-3.30} & \text{for } 10.0 M_\odot < M < 125 M_\odot \\
    0 & \text{otherwise.} 
\end{cases} 
\] (13.156)

Piecewise Power-law

Arbitrary piecewise power-law IMFs can be defined using the PiecewisePowerLaw method. The IMF will be constructed such that:

\[
\phi(M) \propto M^{\alpha_i} \text{ if } M_i \leq M < M_{i+1}, 
\] (13.157)

where \( i = 1 \ldots N \), the \( M_i \) are given by [imfPiecewisePowerLawMassPoints] and the \( \alpha_i \) are given by [imfPiecewisePowerLawExponents]. (Note that [imfPiecewisePowerLawMassPoints] must contain \( N+1 \) elements, while [imfPiecewisePowerLawExponents] contains only \( N \) elements.) The normalization of each power-law piece is chosen to ensure a continuous IMF that is normalized to unit mass overall.

Salpeter

The Salpeter IMF is defined by [Salpeter, 1955]:

\[
\phi(M) \propto \begin{cases} 
    M^{-2.35} & \text{for } 0.1 M_\odot < M < 125 M_\odot \\
    0 & \text{otherwise.} 
\end{cases} 
\] (13.158)
13. Physical Implementations

Scalo

The Scalo IMF is defined by [Scalo, 1986]:

\[ \phi(M) \propto \begin{cases} 
M^{+1.60} & \text{for } 0.10M_\odot < M < 0.18M_\odot \\
M^{-1.01} & \text{for } 0.18M_\odot < M < 0.42M_\odot \\
M^{-2.75} & \text{for } 0.42M_\odot < M < 0.62M_\odot \\
M^{-2.08} & \text{for } 0.62M_\odot < M < 1.18M_\odot \\
M^{-3.50} & \text{for } 1.18M_\odot < M < 3.50M_\odot \\
M^{-2.63} & \text{for } 3.50M_\odot < M < 125M_\odot \\
0 & \text{otherwise.} 
\end{cases} \] (13.159)

13.23. Intergalactic Medium State

The thermal and ionization state of the intergalactic medium is implemented by the algorithm selected using the `intergalacticMediumStateMethod` parameter.

13.23.1. Simple

Selected using `intergalacticMediumStateMethod=simple`, this method provides a simple model of reionization in which the universe is assumed to be fully neutral prior to the redshift given by \([igmStateSimpleReionizationRedshift]\) and fully ionized thereafter. The temperature is given by \([igmStateSimplePreReionizationTemperature]\) before reionization, and \([igmStateSimpleReionizationTemperature]\) thereafter.

13.23.2. RecFast

Selected using `intergalacticMediumStateMethod=RecFast`, this method computes the state of the intergalactic medium using the RecFast code Seager et al. [2000], Wong et al. [2008]. The RecFast code will be downloaded and run to compute the intergalactic medium state as needed, which will then be stored for future use.

13.23.3. File

Selected using `intergalacticMediumStateMethod=file`, this method reads the state of the intergalactic medium from a file and interpolates in the tabulated results. The XML file containing the table should have the following form:

```xml
<igm>
  <electronFraction>
    <datum>1.1590278</datum>
    <datum>1.1590278</datum>
  .
  .
  .
  </electronFraction>
  <hIonizedFraction>
    <datum>1.0000000</datum>
    <datum>1.0000000</datum>
  .
  .
  .
</igm>
```
13.24. Chemical State

The `electronFraction`, `hIonizedFraction`, `heIonizedFraction`, and `matterTemperature` elements should contain `datum` elements listing the relevant quantity for each redshift listed in the `redshift` element.

13.24. Chemical State

The chemical state of gas is implemented by the algorithm selected using the `ionizatonStateMethod` parameter.

13.24.1. Atomic Collisional Ionization Equilibrium Using CLOUDY

Selected using `ionizatonStateMethod=atomic_CIE_Cloudy`, this method computes the chemical state using the CLOUDY code and under the assumption of collisional ionization equilibrium with no molecular contribution. Abundances are Solar, except for zero metallicity calculations which use CLOUDY's “primordial” metallicity. The helium abundance for non-zero metallicity is scaled between primordial and Solar values linearly with metallicity. The CLOUDY code will be downloaded and run to compute the cooling function as needed, which will then be stored for future use. As this process is slow, a precomputed table is provided with GALACTICUS. If metallicities outside the range tabulated in this file are required it will be regenerated with an appropriate range.

13.24.2. Collisional Ionization Equilibrium From File

Selected using `ionizatonStateMethod=CIE_from_file`, in this method the chemical state is read from a file specified by the `chemicalStateFile` parameter. The format of this file is specified in §17.4.1. The chemical state is assumed to be computed under conditions of collisional ionization equilibrium and
therefore densities scale as $\rho$. Optional HI and HII densities, if present in the file, will be read and used when returning the densities of “chemical” species.

13.25. Merger Tree Construction

Merger trees are “constructed\(^5\)” by the method specified by the $\text{mergerTreeConstructMethod}$ parameter.

13.25.1. Read From File

Selected with $\text{mergerTreeConstructMethod} = \text{read}$, this method reads merger tree structures from file by the designated importer (see §13.30).

13.25.2. Build

Selected with $\text{mergerTreeConstructMethod} = \text{build}$, this method first creates a distribution of tree root halo masses and then builds a merger tree using the algorithm specified by the $\text{mergerTreeBuildMethod}$ parameter.

If $\text{[mergerTreeBuildTreesHaloMassDistribution]} = \text{read}$, then these masses will be read from a file specified by $\text{[mergerTreeBuildTreeMassesFile]}$. Otherwise, the root halo masses are selected to range between $\text{mergerTreeBuildHaloMassMinimum}$ and $\text{mergerTreeBuildHaloMassMaximum}$ with $\text{mergerTreeBuildTreesPerDecade}$ trees per decade of root halo mass on average. Trees are rooted at $\text{mergerTreeBuildTreesBaseRedshift}$ and tree building will begin with the $\text{mergerTreeBuildTreesBeginAtTree}^{th}$ tree\(^6\). The root halo masses are usually drawn from the density function specified by the halo mass function sampling density method (see §13.8.6). The distribution of $x$ is determined by the input parameter $\text{mergerTreeBuildTreesHaloMassDistribution}$ with options:

- **uniform** $x$ is distributed uniformly between 0 and 1;
- **quasi** $x$ is distributed using a quasi-random sequence;
- **random** $x$ is distributed using a pseudo-random sequence;
- **read** halos masses are instead read from a file.

In the case of reading root halo masses from a file, the file can be either an XML or HDF5 file. An XML file should have the following form:

```xml
<mergerTrees>
  <treeRootMass>13522377303.5998</treeRootMass>
  <treeRootMass>19579530191.8709</treeRootMass>
  <treeRootMass>21061025282.9613</treeRootMass>
  ...
  ...
  <treeWeight>13522377303.5998</treeWeight>
  <treeWeight>19579530191.8709</treeWeight>
  <treeWeight>21061025282.9613</treeWeight>
  ...
</mergerTrees>
```

\(^5\)By “construct” we mean any process of creating a representation of a merger tree within \textsc{Galacticus}.

\(^6\)This will normally be set to 1 to begin with the first tree. Other values allow to begin on later trees for debugging purposes.
where each `treeRootMass` element gives the mass (in Solar masses) of the root halo of a tree to generate, and the (optional) `treeWeight` elements give the corresponding weight (in units of Mpc\(^{-3}\)) to assign to each tree. An HDF5 file should have the following structure:

```hdf5
HDF5 {
  GROUP '/' {
    DATASET 'treeRootMass' {
      DATATYPE H5T_IEEE_F64BE
      DATASPACE SIMPLE { ( * ) / ( * ) }
    }
    DATASET 'treeWeight' {
      DATATYPE H5T_IEEE_F64BE
      DATASPACE SIMPLE { ( * ) / ( * ) }
    }
  }
}
```

where the `treeRootMass` dataset contains the mass (in Solar masses) of the root halo of a tree to generate, and the (optional) `treeWeight` dataset contains the weight (in units of Mpc\(^{-3}\)) to assign to each tree.

### 13.25.3. Fully Specified

Selected with `mergerTreeConstructMethod=fullySpecified`, this method will construct a merger tree, and set properties of components in each node, using a description read from an XML document. The document is specified via the `[mergerTreeConstructFullySpecifiedFileName]` input parameter.

The tree specification document looks as follows:

```xml
<!-- Simple initial conditions test case -->
<initialConditions>
  <node>
    <index>2</index>
    <parent>1</parent>
    <firstChild>-1</firstChild>
    <sibling>-1</sibling>
    <basic>
      <time>1.0</time>
      <timeLastIsolated>1.0</timeLastIsolated>
      <mass>1.0e12</mass>
      <accretionRate>7.9365079e9</accretionRate>
    </basic>
    <spin>
      <spin>0.1</spin>
    </spin>
    <disk>
      <massGas>1.0e10</massGas>
      <angularMomentum>1.0e10</angularMomentum>
    </disk>
  </node>
</initialConditions>
```
13. Physical Implementations

The document consists of a set of node elements, each of which defines a single node in the merger tree. Each node element must specify the index of the node, along with the index of the node’s parent, firstChild, and sibling.

Each node element may contain elements which specify the properties of a component in the node. For example, a basic element will specify properties of the “basic” component. If multiple elements for a given component type are present, then multiple instances of that component will be created in the node.

Within a component definition element scalar properties are set using an element with the same name as that property (e.g. mass in the basic components in the above example). Rank-1 properties are set using a list of elements with the same name as the property (e.g. position in the position component in the above example).

For composite properties (e.g. abundances), the specification element should contain sub-elements that specify each property of the composite. Currently only the abundances object supports specification in this way, as detailed below:

abundances (See abundancesGas in the above example.) The total metal content is specified via a metals element. If other elements are being tracked, their content is specified via an element with the short-name of the element (e.g. Fe for iron).

13.25.4. Smooth Accretion

Selected with mergerTreeConstructMethod=smoothAccretion, this method builds a branchless merger tree with a smooth accretion history using the selected mass accretion history method (see §17.4.1). The tree has a final mass of mergerTreeHaloMass (in units of $M_\odot$) at redshift mergerTreeBaseRedshift and
is continued back in time by decreasing the halo mass by a factor \texttt{mergerTreeHaloMassDeclineFactor} at each new node until a specified \texttt{mergerTreeHaloMassResolution} (in units of $M_\odot$) is reached.

### 13.25.5. State Restore

Selected with \texttt{mergerTreeConstructMethod=stateRestore}, this method will restore a merger tree whose complete internal state was written to file. It is intended primarily for debugging purposes to allow a tree to begin processing just prior to the point of failure. To use this method, the following procedure should be followed:

1. Identify a point in the evolution of the tree suitably close to, but before, the point of failure;

2. Insert appropriate code into \texttt{GALACTICUS} to have it call the function to store the state of the file and then stop, e.g.:

   ```
   use Merger_Trees_State_Store
   
   if (<conditions are met>) then
     call Merger_Trees_State_Store(thisTree,’storedTree.dat’)
     stop ’tree internal state was stored’
   end if
   ```

3. Run the model ensuring that \texttt{[stateFileRoot]} is set to a suitable file root name to allow the internal state of \texttt{GALACTICUS} to be stored;

4. Remove the code inserted above and recompile;

5. Run \texttt{GALACTICUS} with an input parameter file identical to the one used previously except with \texttt{[mergerTreeConstructMethod]=stateRestore}, \texttt{[stateFileRoot]} removed, \texttt{[stateRetrieveFileRoot]} set to the value previously used for \texttt{[stateFileRoot]} and \texttt{[mergerTreeStateStoreFile]=storedTree.dat}.

This should restore the tree and the internal state of \texttt{GALACTICUS} precisely from the point where they were saved and produce the same subsequent evolution.

Note that currently this method does not support storing and restoring of trees which contain components that have more than one instance.

### 13.26. Merger Tree Building Mass Resolution

The method to be used for computing the mass resolution to use when building merger trees is specified by the \texttt{mergerTreesBuildMassResolutionMethod} parameter.

#### 13.26.1. Fixed

Selected with \texttt{mergerTreesBuildMassResolutionMethod=fixed}, this method assumes a fixed mass resolution of \texttt{[mergerTreeBuildMassResolutionFixed]} for all merger trees.
13. Physical Implementations

13.26.2. Scaled

Selected with `mergerTreesBuildMassResolutionMethod=scaled`, this method computes the mass resolution to be the larger of `[mergerTreeBuildMassResolutionScaledMinimum]` and `[mergerTreeBuildMassResolutionScaledFraction] \times M_{\text{base}}`, where $M_{\text{base}}$ is the base mass of the merger tree.

13.27. Merger Tree Branching

The method to be used for computing branching probabilities in merger trees is specified by the `treeBranchingMethod` parameter.

13.27.1. Modified Press-Schechter

Selected with `treeBranchingMethod=modifiedPress-Schechter`, this method uses the algorithm of Parkinson et al. [2008] to compute branching ratios. The parameters $G_0$, $\gamma_1$ and $\gamma_2$ of their algorithm are specified by the input parameters `modifiedPressSchechterG0`, `modifiedPressSchechterGamma1` and `modifiedPressSchechterGamma2` respectively. Additionally, the parameter `modifiedPressSchechterFirstOrderAccuracy` limits the step in $\delta_{\text{crit}}$ so that it never exceeds $\text{modifiedPressSchechterFirstOrderAccuracy} \sqrt{2}[\sigma^2(M_2/2) - \sigma^2(M_2)]$, which ensures the first order expansion of the merging rate that is assumed is accurate.

13.27.2. Generalized Press-Schechter

Selected with `treeBranchingMethod=generalizedPress-Schechter`, this method computes branching probabilities from solutions to the excursion set barrier first crossing rate problem (using the selected `excursionSetFirstCrossingMethod`; see §17.4.1). Specifically, the branching probability per unit time is:

$$\frac{df}{dt} = \frac{dt}{d\omega} \int_{M_{\text{min}}}^{M/2} M \frac{df}{d\omega} \frac{dS}{dM'} \frac{dt}{d\omega} G[\omega, \sigma(M), \sigma(M')] \, dM', \tag{13.160}$$

where $\omega = \delta_{\text{crit}} / D(t)$. The rate of accretion of mass in halos below the resolution limit of the merger tree is

$$\frac{dR}{dt} = \frac{dt}{d\omega} \int_{0}^{M_{\text{min}}} \frac{df}{d\omega} \frac{dS}{dM'} \frac{dt}{d\omega} G[\omega, \sigma(M), \sigma(M')] \, dM'. \tag{13.161}$$

In the above, $G[\omega, \sigma(M), \sigma(M')]$ is a modification to the merger rate as computed by the selected `treeBranchingModifierMethod` (see §17.4.1). If `[generalizedPressSchechterSmoothAccretion]=true` then smooth accretion (i.e. accretion of matter not in dark matter halos) is accounted for at the rate:

$$\frac{dR_s}{dt} = \frac{dt}{d\omega} G[\omega, \sigma_{\text{max}}, \sigma(M')] \frac{df}{dt}, \tag{13.162}$$

where $\sigma_{\text{max}}$ is the peak value of $\sigma(M)$ (for the lowest mass halos) and $d\tilde{f} / dt$ is the rate at which excursion set trajectories fail to cross the barrier on any mass scale.

13.28. Merger Tree Branching Modifier

The method to be used for modifying branching probabilities in merger trees is specified by the `treeBranchingModifierMethod` parameter.
13.28.1. Null

Selected with `treeBranchingModifierMethod=null`, this method makes no change to the branching probability.


Selected with `treeBranchingModifierMethod=Parkinson-Cole-Helly2008`, this method modifies branching probabilities according to the prescription of Parkinson et al. [2008]. Specifically, the branching probability is multiplied by:

\[
G(\delta_p, \sigma_c, \sigma_p) = G_0 \left( \frac{\sigma_p}{\sigma_p} \right)^{\gamma_1} \left( \frac{\delta_p}{\sigma_p} \right)^{\gamma_2}
\]

where \( \delta_p \) is the current critical overdensity for collapse for the parent halo, and \( \sigma_c \) and \( \sigma_p \) are the root-variance of the smooth mass-density field on scales corresponding to the masses of child and parent halos respectively. The parameters of the fit can be adjusted via input parameters: \( G_0 = [\text{modifiedPressSchechterG0}] \), \( \gamma_1 = [\text{modifiedPressSchechterGamma1}] \), and \( \gamma_2 = [\text{modifiedPressSchechterGamma2}] \).

13.29. Merger Tree Building

The method to be used for building merger trees is specified by the `mergerTreeBuildMethod` parameter.


Selected with `mergerTreeBuildMethod=Cole2000`, this method uses the algorithm described by Cole et al. [2000], with a branching probability method selected via the `treeBranchingMethod` parameter. This action of this algorithm is controlled by the following parameters:

- `mergerTreeBuildCole2000MergeProbability` The maximum probability for a binary merger allowed in a single timestep. This allows the probability to be kept small, such the the probability for multiple mergers within a single timestep is small.

- `mergerTreeBuildCole2000AccretionLimit` The maximum fractional change in mass due to sub-resolution accretion allowed in any given timestep when building the tree.

- `mergerTreeBuildCole2000HighestRedshift` The highest redshift to which the tree should be built. Any branch reaching this redshift will be terminated. Typically this should be set to a high value such that branches terminate when the resolution limit it reached, but specifying a maximum redshift can be useful in some situations.

The minimum halo mass that the algorithm will follow is determined by the selection merger tree building mass resolution method (see §13.26). Mass accretion below this scale is treated as smooth accretion and branches are truncated once they fall below this mass.

13.30. Merger Tree Importing

When merger trees are to be read from file, a number of different file formats are supported. An “importer” class is used to read these files and place the contents into internal data structures that GALACTICUS can then manipulate. The importer class to use is specified by the `[mergerTreeImporterMethod]` parameter.
13. Physical Implementations

13.30.1. Galacticus

Selected with `mergerTreeImporterMethod=galacticus`, this method reads merger tree structures from an HDF5 file specified by the `mergerTreeReadFileName` parameter. The structure of these HDF5 files is described in §6.2.

13.30.2. Sussing Merger Trees

Selected with `mergerTreeImporterMethod=sussing`, this method reads merger tree structures from files following the format designed for the “Sussing Merger Trees” workshop [Srisawat et al., 2013], along with Amiga’s Halo Finder (AHF) format halo catalogs. A descriptor file must be specified via the `mergerTreeReadFileName` parameter. This descriptor file should have the following format:

```plaintext
simulation.txt
MergerTree+AHF.txt
snapidzred.txt
AHF/62.5_dm_000.z50.000.AHF_halos
AHF/62.5_dm_001.z30.000.AHF_halos
AHF/62.5_dm_002.z19.916.AHF_halos
.
.
AHF/62.5_dm_061.z0.000.AHF_halos
```

in which each line specifies a file to be read (by default path names are relative to the location of the descriptor file—fully-qualified path names can also be given).

The first line identifies a file which specifies properties of the simulation. This file should look like:

```plaintext
WMAP7 cosmology:
----------------
Omega0 = 0.272
OmegaLambda0 = 0.728
h = 0.704
```

The second line identifies the merger tree file which must be in the format specified by Srisawat et al. [2013].

The third line of the descriptor file specifies a snapshot file which should have the following format:

```plaintext
# snapnum a z t(t0) t(year)
0 0.0196080 49.9996 0.00354284 4.87485e+07
1 0.0322580 30.0001 0.00747572 1.02864e+08
2 0.0478110 19.9157 0.0134888 1.85602e+08
3 0.0519650 18.2437 0.0152842 2.10306e+08
4 0.0564190 16.7245 0.0172905 2.37912e+08
5 0.0611880 15.3431 0.0195280 2.68700e+08
```

Currently only the cosmological parameter and box length are read from this file.

The second line identifies the merger tree file which must be in the format specified by Srisawat et al. [2013].

The third line of the descriptor file specifies a snapshot file which should have the following format:
13.31. Merger Tree Pre-evolution Processing

This file must contain one line for each snapshot of the simulation, giving the snapshot number, expansion factor, redshift, fractional time (relative to present day), and age of the universe (in years).

Subsequent lines identify the AHF halo files for each snapshot (files can be listed in any order).

Merger tree files of this type can be split into subvolumes before processing. This is useful if the file is too large to read into memory in one go. The number of subvolumes to use (in each of the three dimensions of the simulation cube) is specified by the [mergerTreeImportSussingSubvolumeCount] parameter. The specific subvolume to process is specified by the [mergerTreeImportSussingSubvolumeIndex] parameter, which should give the index (running from 0 to [mergerTreeImportSussingSubvolumeCount]−1) in each dimension (whitespace separated). To ensure that no halos are missed from trees near the edge of the subvolume, a buffer region around the subvolume is also read. The width of this buffer (in units of Mpc/h to follow the format convention) is specified via the [mergerTreeImportSussingSubvolumeBuffer] parameter.

13.31. Merger Tree Pre-evolution Processing

Arbitrary processing of merger trees prior to their evolution can be carried out using the mergerTreePreEvolveTask directive (see §17.4.3). Currently defined tasks are defined below.

13.31.1. Enforce Monotonic Mass Growth

This task enforces monotonic growth a halo mass along each branch of each merger tree. It does this by searching the tree for nodes which are less massive than the sum of the masses of their immediate progenitors, and increasing the mass of such nodes to equal the sum of the masses of their immediate progenitors. To enforce monotonic mass growth along branches set [mergerTreeEnforceMonotonicGrowth]=true.

13.31.2. Interpolate Tree to Time Grid

This task will interpolate the merger tree structure onto a new array of timesteps if [mergerTreeRegridTimes]=true. The timestep array is specified via the parameters:

[mergerTreeRegridSpacing] The spacing of the timesteps. Five options are available: linear will space timesteps uniformly in expansion factor, logarithmic will space timesteps uniformly in the logarithm of expansion factor, log critical density will space timesteps uniformly in the logarithm of critical density, δₖ, millennium will use times corresponding to the redshifts of snapshots in the Millennium Simulation database, while read will use times corresponding to the redshifts specified in the mergerTreeRegridRedshifts parameter;

[mergerTreeRegridStartExpansionFactor] The smallest expansion factor in the array (ignored for millennium and read spacings);

[mergerTreeRegridEndExpansionFactor] The largest expansion factor in the array (ignored for millennium and read spacings);

[mergerTreeRegridCount] The number of timesteps in the array;

Along each branch of the tree, new halos are inserted at times corresponding to the times in the resulting array. The masses of these nodes are linearly interpolated between the existing nodes on the branch.
Once these new nodes have been added, all other nodes are removed from the tree. The processing is useful to construct representations of trees as they would be if only sparse time sampling were available. As such, it is useful for exploring how the number of snapshots in merger trees extracted from N-body simulations affects the properties of galaxies that form in them.

13.31.3. Mass Accretion History Output

Output of the mass accretion history (i.e. the mass of the node on the primary branch as a function of time) for each merger tree can be requested by setting \( \text{massAccretionHistoryOutput} = \text{true} \). If requested, an additional group, massAccretionHistories, is made in the GALACTICUS output file. This group will contain a subgroup for each merger tree (mergerTreeN where N is the merger tree index) within which three datasets, nodeId, nodeTime and nodeMass, can be found. These give the index, time and mass of the node on the primary branch of the tree at all times for which the tree is defined.

13.31.4. Tree Pruning By Mass

This task allows for branches of merger trees to be pruned—i.e. nodes below a specified mass limit are removed from the tree prior to any evolution. This can be useful for convergence studies for example. To prune branches set \( \text{mergerTreePruneBranches} = \text{true} \) and set \( \text{mergerTreePruningMassThreshold} \) to the desired mass threshold below which nodes will be pruned.

13.31.5. Tree Pruning By Hierarchy

This task allows for branches of merger trees to be pruned by hierarchy—i.e. nodes below a given depth in the hierarchy are removed from the tree prior to any evolution. To prune branches by hierarchy depth set \( \text{mergerTreePruneHierarchyAtDepth} \) to the desired depth at which to prune. For example, a value of 1 will result in all branches except for the main branch being removed, while a value of 2 will remove all branches that do not merge directly onto the main branch. (Note that setting \( \text{mergerTreePruneHierarchyAtDepth} \) to zero will result in no pruning.)

13.31.6. Tree Pruning By Essential Node

This task allows for branches of merger trees to be pruned to leave only the branch occupied by an “essential” node, along with any progenitor branches of that node. This is intended primarily for debugging purposes where it can be convenient to cut away all non-essential branches of the tree. This pruning task is activated by setting \( \text{mergerTreePruneNonEssential} = \text{true} \). The essential node is specified by \( \text{mergerTreePruningNonEssentialID} \) and \( \text{mergerTreePruningNonEssentialTime} \), which specify the index of the node, and the time at which its properties are required. Specifying the time is important—if the node is a satellite at this time, then the pruning will not remove any progenitors of the parent node in which the essential node lives at the specified time.

13.32. Chemical Reaction Rates

Methods for computing chemical reaction rates are selected via the \( \text{chemicalReactionRatesMethods} \) parameter. Multiple methods can be selected—their rates are cumulated.

---

7 The base node of the tree is never removed, even if it does not lie on one of the times in the constructed array.
8 The main branch is defined as depth 0. Other branches are assigned a depth equal to the depth of the branch onto which they merge plus 1. For example, any branch which merges directly onto the main branch is defined as depth 1.
13.33. Ram Pressure Induced Mass Loss Rates in Disks/Spheroids

13.33.1. Null
Selected with \texttt{chemicalReactionRatesMethods=null} this is a null method which does not set any rates.

13.33.2. Hydrogen Network
Selected with \texttt{chemicalReactionRatesMethods=hydrogenNetwork} this method computes rates using the network of reactions and fitting functions from Abel et al. [1997] and Tegmark et al. [1997]. The parameter \texttt{[hydrogenNetworkFast]} controls the approximations made. If set true then H\textsuperscript{−} is assumed to be at equilibrium abundance, H\textsuperscript{2} reactions are ignored and other slow reactions are ignored (see Abel et al. 1997).

13.33. Ram Pressure Induced Mass Loss Rates in Disks/Spheroids

The methods for computing ram pressure induced rates of mass loss in disks/spheroids are selected via the \texttt{ramPressureStrippingMassLossRate(Disks|Spheroids)Method} parameter.

13.33.1. Simple
Selected with \texttt{ramPressureStrippingMassLossRate(Disks|Spheroids)Method=simple} this method computes the mass loss rate to be, for disks:

\[
\dot{M}_{\text{gas}, \text{disk}} = \min \left( \frac{F_{\text{hot}, \text{host}}}{2\pi G \Sigma_{\text{gas}}(r_{\text{half}}) \Sigma_{\text{total}}(r_{\text{half}})}, \frac{R_{\text{maximum}}}{\tau_{\text{dyn, disk}}} \right) \frac{M_{\text{gas, disk}}}{\tau_{\text{dyn, disk}}}, \tag{13.164}
\]

where \(F_{\text{hot, host}}\) is the ram pressure force due to the hot halo of the node’s host (computed using the selected hot halo ram pressure force method; see §13.18), \(\Sigma_{\text{gas}}(r)\) is the gas surface density in the disk, \(\Sigma_{\text{total}}(r)\) is the total surface density in the disk, \(r_{\text{half}}\) is the disk half-mass radius, \(M_{\text{gas, disk}}\) is the total gas mass in the disk, \(\tau_{\text{dyn, disk}} = r_{\text{disk}}/v_{\text{disk}}\) is the dynamical time in the disk, and \(R_{\text{maximum}} = \texttt{[ramPressureStrippingMassLossRateDisksSimpleFractionalRateMaximum]}\) controls the maximum allowed rate of mass loss. For spheroids the mass loss rate is:

\[
\dot{M}_{\text{gas}} = -\max(\alpha, R_{\text{maximum}}) M_{\text{gas}}/\tau_{\text{spheroid}}, \tag{13.165}
\]

where \(R_{\text{maximum}} = \texttt{[ramPressureStrippingMassLossRateSpheroidSimpleFractionalRateMax]}\)

\[
\alpha = \frac{F_{\text{hot, host}}}{F_{\text{gravity}}}, \tag{13.166}
\]

and,

\[
F_{\text{gravity}} = \frac{4}{3} \rho_{\text{gas}}(r_{1/2}) \frac{GM_{\text{total}}(r_{1/2})}{r_{1/2}} \tag{13.167}
\]

is the gravitational restoring force in the spheroid at the half-mass radius, \(r_{1/2}\) [Takeda et al., 1984].

13.33.2. Null
Selected with \texttt{ramPressureStrippingMassLossRate(Disks|Spheroids)Method=null} this method assumes zero mass loss rate always.

13.34. Satellite Dynamical Friction

The method to be used for computing the satellite vector acceleration due to dynamical friction is specified by the \texttt{satelliteDynamicalFrictionMethod} parameter.
13. Physical Implementations

13.34.1. Null

Selected with `satelliteDynamicalFrictionMethod=null`, this method assumes that there is no dynamical friction in the system.

13.34.2. Chandrasekhar (1943)

Selected with `satelliteDynamicalFrictionMethod=Chandrasekhar1943`, this method uses the Chandrasekhar [1943] formula to compute the acceleration of a satellite at radius $r$ from the center of the host due to dynamical friction:

$$a_{DF} = -\frac{4\pi G^2 M_{\text{sat}} \rho_{\text{host}}(r)}{v_{\text{sat}}^3} \ln \Lambda \left[ \text{erf}(x) - \frac{2x}{\sqrt{\pi}} \exp(-x^2) \right] v_{\text{sat}},$$  \hspace{1cm} (13.168)

where $M_{\text{sat}}$ and $v_{\text{sat}}$ are the satellite’s mass and velocity, respectively, $v_{\text{sat}} = |v_{\text{sat}}|$, $\rho_{\text{host}}(r)$ is the host’s density profile, $\ln \Lambda = \text{satelliteDynamicalFrictionChandrasekharCoulombLogarithm}$ is the Coulomb logarithm, and $x \equiv v_{\text{sat}}/\sqrt{2}\sigma(r)$, where $\sigma(r)$ is the velocity dispersion of the host halo at radius $r$, approximated to be equal to the host virial velocity, $v_{\text{vir}}$.

13.35. Satellite Tidal Fields

The method for computing the tidal field experienced by satellite galaxies is selected via `satellitesTidalFieldMethod`.

13.35.1. Null

Selected with `satellitesTidalFieldMethod=null` this method assumes a zero tidal field always.

13.35.2. Spherical Symmetry

Selected with `satellitesTidalFieldMethod=sphericalSymmetry` this method assumes a spherically-symmetric host halo, and computes the tidal field accordingly using:

$$F = \frac{GM_{\text{host}}(<r_p)}{r_p^3} - 4\pi G \rho_{\text{host}}(r_p) + \omega_p^2,$$  \hspace{1cm} (13.169)

where $r_p$ is the pericentric radius, $M_{\text{host}}(<r)$ is the mass of the host halo enclosed within a sphere of radius $r$, $\rho_{\text{host}}(r)$ is the host density at radius $r$, and $\omega_p$ is the orbital angular velocity at pericenter.

13.36. Satellite Tidal Stripping

The method to be used for computing the satellite mass loss rate due to tidal stripping is specified by the `satelliteTidalStrippingMethod` parameter.

13.36.1. Null

Selected with `satelliteTidalStrippingMethod=null`, this method assumes that there is no tidal stripping in the system.
13.36.2. Zentner (2005)

Selected by default with \texttt{satelliteTidalStrippingMethod=Zentner2005}, this method uses the formalism of Zentner et al. [2005] to compute the mass loss rate $\dot{M}_\text{sat}$:

$$\dot{M}_\text{sat} = -\alpha \frac{M_\text{sat}(>r_\text{tidal})}{T_\text{orb}},$$

(13.170)

where $\alpha = \texttt{[satelliteTidalStrippingZentnerRate]}$, $T_\text{orb}$ is the orbital period of the satellite, and $r_\text{tidal}$ is the tidal radius of the satellite, given by the King [1962] formula:

$$r_\text{tidal} = \left(\frac{GM_\text{sat}}{\omega^2 - d^2 \Phi/dr^2}\right)^{1/3},$$

(13.171)

where $\omega$ is the orbital angular velocity of the satellite, and $\Phi(r)$ is the gravitational potential due to the host.

13.37. Satellite Tidal Heating

The method to be used for computing the integrated, normalized (i.e. the energy divided by radius squared) tidal heating energy, $Q_\text{tidal}$, for subhalos is specified by the \texttt{satelliteTidalHeatingMethod} parameter.

13.37.1. Null

Selected with \texttt{satelliteTidalHeatingMethod=null}, this method assumes that there is no tidal heating.


Selected by default with \texttt{satelliteTidalHeatingMethod=Gnedin1999}, this method uses the formalism of Gnedin et al. [1999] to compute the heating rate:

$$Q_\text{tidal} = \frac{1}{3} \epsilon \left[1 + \left(\frac{T_\text{shock}}{T_\text{orb}}\right)^2\right]^{-\gamma} g_{ij} G_{ij}$$

(13.172)

where $T_\text{orb}$ and $T_\text{shock}$ are the orbital period and shock duration, respectively, of the satellite, $\epsilon = \texttt{[satelliteTidalHeatingGnedinEpsilon]}$ and $\gamma = \texttt{[satelliteTidalHeatingGnedinGamma]}$ are model parameters, $g_{ij}$ is the tidal tensor, and $G_{ij}$ is the integral with respect to time of $g_{ij}$ along the orbit of the satellite. Upon tidal heating, a mass element at radius $r_i$ expands to radius $r_f$, according to the equation

$$\frac{1}{r_f} = \frac{1}{r_i} - \frac{2\gamma^3 Q_\text{tidal}}{GM_\text{sat}(<r_i)},$$

(13.173)

13.38. Star Formation Rate Surface Densities

The method for computing surface densities of star formation rate in disks is selected via \texttt{starFormationRateSurfaceDensityDisksMethod}.
13. Physical Implementations

13.38.1. Kennicutt-Schmidt

Selected with starFormationRateSurfaceDensityDisksMethod=Kennicutt-Schmidt this method assumes that the Kennicutt-Schmidt law holds [Schmidt, 1959, Kennicutt, 1998]:

$$\dot{\Sigma}_* = A \left( \frac{\Sigma_{H \text{I}}}{M_\odot \text{pc}^{-2}} \right)^N,$$

(13.174)

where $A = \text{[starFormationKennicuttSchmidtNormalization]}$ and $N = \text{[starFormationKennicuttSchmidtExponent]}$ are parameters. Optionally, if the [starFormationKennicuttSchmidtTruncate] parameter is set to true, then the star formation rate is truncated below a critical surface density such that

$$\dot{\Sigma}_* = \begin{cases} A \left( \frac{\Sigma_{H \text{I}}}{M_\odot \text{pc}^{-2}} \right)^N & \text{if } \Sigma_{\text{gas,disk}} > \Sigma_{\text{crit}} \\ A \left( \frac{\Sigma_{H \text{I}}}{M_\odot \text{pc}^{-2}} \right)^N (\Sigma_{\text{gas,disk}}/\Sigma_{\text{crit}})^\alpha & \text{otherwise} \end{cases}$$

(13.175)

Here, $\alpha = \text{[starFormationKennicuttSchmidtExponentTruncated]}$ and $\Sigma_{\text{crit}}$ is a critical surface density for star formation which we specify as

$$\Sigma_{\text{crit}} = \frac{q_{\text{crit}} \kappa \sigma_{\text{gas}}}{\pi G},$$

(13.176)

where $\kappa$ is the epicyclic frequency in the disk, $\sigma_{\text{gas}}$ is the velocity dispersion of gas in the disk and $q_{\text{crit}} = \text{[toomreParameterCritical]}$ is a dimensionless constant of order unity which controls where the critical density occurs. We assume that $\sigma_{\text{gas}}$ is a constant equal to [velocityDispersionDiskGas] and that the disk has a flat rotation curve such that $\kappa = \sqrt{2V/R}$.

13.38.2. Extended Schmidt

Selected with starFormationRateSurfaceDensityDisksMethod=extendedSchmidt this method assumes that the extended Schmidt law holds [Shi et al., 2011]:

$$\dot{\Sigma}_* = A \left( \frac{\Sigma_{\text{gas}}}{M_\odot \text{pc}^{-2}} \right)^{N_1} \left( \frac{\Sigma_*}{M_\odot \text{pc}^{-2}} \right)^{N_2},$$

(13.177)

where $A = \text{[starFormationExtendedSchmidtNormalization]}$, $N_1 = \text{[starFormationExtendedSchmidtGasExponent]}$ and $N_2 = \text{[starFormationExtendedSchmidtStarExponent]}$ are parameters.

13.38.3. Blitz-Rosolowsky

Selected with starFormationRateSurfaceDensityDisksMethod=Blitz-Rosolowsky2006 this method assumes that the star formation rate is given by [Blitz and Rosolowsky, 2006]:

$$\dot{\Sigma}_*(R) = \nu_{\text{SF}}(R) \Sigma_{\text{H_2,disk}}(R),$$

(13.178)

where $\nu_{\text{SF}}$ is a frequency given by

$$\nu_{\text{SF}}(R) = \nu_{\text{SF,0}} \left[ 1 + \left( \frac{\Sigma_{\text{H}_2}}{\Sigma_0} \right)^q \right],$$

(13.179)

where $q = \text{[surfaceDensityExponentBlitzRosolowsky]}$ and $\Sigma_0 = \text{[surfaceDensityCriticalBlitzRosolowsky]}$ are parameters and the surface density of molecular gas $\Sigma_{\text{H}_2} = (P_{\text{ext}}/R_0)^{\alpha} \Sigma_{\text{H}_I}$, where $\alpha = \text{[pressureExponentBlitzRosolowsky]}$.
and \( P_0 = \text{[pressureCharacteristicBlitzRosolowsky]} \) are parameters and the hydrostatic pressure in the disk plane assuming location isothermal gas and stellar components is given by

\[
P_{\text{ext}} \approx \frac{\pi}{2} G \Sigma_{\text{gas}} \left[ \Sigma_{\text{gas}} + \left( \frac{\sigma_{\text{gas}}}{\sigma_\star} \right) \Sigma_\star \right]
\]  

(13.180)

where we assume that the velocity dispersion in the gas is fixed at \( \sigma_{\text{gas}} = \text{[velocityDispersionDiskGas]} \) and, assuming \( \Sigma_\star \gg \Sigma_{\text{gas}} \), we can write the stellar velocity dispersion in terms of the disk scale height, \( h_\star \), as

\[
\sigma_\star = \sqrt{\pi G h_\star \Sigma_\star}
\]

(13.181)

where we assume \( h_\star / R_{\text{disk}} = \text{[heightToRadialScaleDiskBlitzRosolowsky]} \).

13.39. Star Formation Timescales

The methods for computing star formation timescales in disks and spheroids are selected via the \text{starFormationTimescaleDisksMethod} and \text{starFormationTimescaleSpheroidsMethod} respectively.

13.39.1. Fixed

Selected with \text{starFormationTimescaleDisksMethod=fixed} this method assumes a fixed timescale for star formation \text{[starFormationTimescaleDisksFixedTimescale]} (in Gyr).
13. Physical Implementations

13.39.2. Halo Scaling
Selected with starFormationTimescaleDisksMethod=haloScaling this method assumes a timescale for star formation of

\[ \tau_\star = \tau_{\star,0} \left( \frac{V_{\text{vir}}}{200 \text{km/s}} \right)^{\alpha_*} (1 + z)^{\beta_*}, \]  \hspace{1cm} (13.188)

where \( \tau_{\star,0} = \text{[starFormationTimescaleDisksHaloScalingTimescale]} \), \( \alpha_* = \text{[starFormationTimescaleDisksHaloScalingVirialVelocityExponent]} \) and \( \beta_* = \text{[starFormationTimescaleDisksHaloScalingRedshiftExponent]} \).

13.39.3. Dynamical Time
Selected with starFormationTimescale[Disks|Spheroids]Method=dynamicalTime this method computes the star formation timescale to be:

\[ \tau_\star = \epsilon_{\star}^{-1} \tau_{\text{dynamical}} \left( \frac{V}{200 \text{km/s}} \right)^{\alpha_*}, \]  \hspace{1cm} (13.189)

where \( \epsilon_{\star} = \text{[starFormation[Disks|Spheroids]Efficiency]} \) and \( \alpha_* = \text{[starFormation[Disks|Spheroids]VelocityExponent]} \) are input parameters, \( \tau_{\text{dynamical}} \equiv r/V \) is the dynamical timescale of the component and \( r \) and \( V \) are the characteristic radius and velocity respectively of the component. The timescale is not allowed to fall below a minimum value specified by starFormation[Disks|Spheroids]MinimumTimescale (in Gyr).

13.39.4. Integrated Surface Density
Selected with starFormationTimescaleDisksMethod=integratedSurfaceDensity this method computes the star formation timescale to be:

\[ \tau_\star = \frac{M_{\text{cold}}}{\int_0^\infty 2\pi r \dot{\Sigma}_\star(r) dr}. \]  \hspace{1cm} (13.190)

where \( \dot{\Sigma}_\star(r) \) is the surface density of star formation rate (see §13.38).

Selected with starFormationTimescaleDisksMethod=Baugh2005 this method assumes that the star formation rate is given by a modified version of the Baugh et al. [2005] prescription:

\[ \tau_\star = \tau_0 (V_{\text{disk}}/200 \text{km/s})^{\alpha a^\beta}, \]  \hspace{1cm} (13.191)

where \( \tau_0 = \text{[starFormationDiskTimescale]} \), \( \alpha = \text{[starFormationDiskVelocityExponent]} \) and \( \beta = \text{[starFormationExpansionExponent]} \).

13.40. Stellar Population Properties

Algorithms for determining stellar population properties—essentially the rates of change of stellar and gas mass and abundances given a star formation rate and fuel abundances (and perhaps a historical record of star formation in the component)—are selected by the stellarPopulationPropertiesMethod parameter.
13.40.1. Instantaneous

Selected with `stellarPopulationPropertiesMethod=instantaneous` this method uses the instantaneous recycling approximation. Specifically, given a star formation rate $\phi$, this method assumes a rate of increase of stellar mass of $M_* = (1 - R)\phi$, a corresponding rate of decrease in fuel mass. The rate of change of the metal content of stars follows from the fuel metallicity, while that of the fuel changes according to

$$\dot{M}_{\text{fuel},Z} = -(1 - R)Z_{\text{fuel}}\phi + p\phi. \quad (13.192)$$

In the above $R$ is the instantaneous recycled fraction and $p$ is the yield, both of which are supplied by the IMF subsystem. The rate of energy input from the stellar population is computed assuming that the canonical amount of energy from a single stellar population (as defined by the `feedbackEnergyInputAtInfinityCanonical`) is input instantaneously.

13.40.2. Noninstantaneous

Selected with `stellarPopulationPropertiesMethod=noninstantaneous` this method assumes fully non-instantaneous recycling and metal enrichment. Recycling and metal production rates from simple stellar populations are computed, for any given IMF, from stellar evolution models. The rates of change are then:

$$\dot{M}_* = \phi - \int_0^t \phi(t')\dot{R}(t - t'; Z_{\text{fuel}}[t'])\,dt', \quad (13.193)$$

$$\dot{M}_{\text{fuel}} = -\phi + \int_0^t \phi(t')\dot{R}(t - t'; Z_{\text{fuel}}[t])\,dt', \quad (13.194)$$

$$\dot{M}_{*,Z} = Z_{\text{fuel}}\phi - \int_0^t \phi(t')Z_{\text{fuel}}(t')\dot{R}(t - t'; Z_{\text{fuel}}[t'])\,dt', \quad (13.195)$$

$$\dot{M}_{\text{fuel},Z} = -Z_{\text{fuel}}\phi + \int_0^t \phi(t')\{Z_{\text{fuel}}(t')\dot{R}(t - t'; Z_{\text{fuel}}[t']) + \dot{p}(t - t'; Z_{\text{fuel}}[t'])\}\,dt', \quad (13.196)$$

where $\dot{R}(t; Z)$ and $\dot{p}(t; Z)$ are the recycling and metal yield rates respectively from a stellar population of age $t$ and metallicity $Z$. The energy input rate is computed self-consistently from the star formation history.

13.41. Stellar Population Spectra

Stellar population spectra are used to construct integrated spectra of galaxies. The method used to compute such spectra is specified by the `stellarPopulationSpectraMethod` parameter.


Selected with `stellarPopulationSpectraMethod=Conroy-White-Gunn2009` this method uses v2.2 of the FSPS code of Conroy et al. [2009] to compute stellar spectra. If necessary, the FSPS code will be downloaded, patched and compiled and run to generate spectra. These tabulations are then stored to file for later retrieval. The file name used is `data/SSP_Spectra_Conroy-et-al_v2.2_imf<imfDescriptor>.hdf5` where `<imfDescriptor>` is the IMF descriptor defined by the selected IMF (see §17.4.3).
13.41.2. File

Selected with `stellarPopulationSpectraMethod=file` this method reads stellar population spectra from an HDF5 file, with format described in §17.4.1.

13.42. Stellar Population Spectra Postprocessing

Stellar population spectra are postprocessed (to handle, for example, absorption by the IGM).

Different chains of postprocessing methods can be applied to different filters. The `[luminosityPostprocessSet]` parameter specifies, for each filter, which chain of postprocessing filters to use. (If this parameter is not present then "default" is assumed for all filters.) The filters used in each chain are specified by the input parameter `[stellarPopulationSpectraPostprocess<setName>Method]` where `<setName>` is the name specified in `[luminosityPostprocessSet]`.

13.42.1. Inoue, Shimizu & Iwata (2014) IGM Attenuation

Selected with `stellarPopulationSpectraPostprocess<setName>Method=inoue2014` this method postprocesses spectra through absorption by the IGM using the results of Inoue et al. [2014].


Selected with `stellarPopulationSpectraPostprocess<setName>Method=meiksin2006` this method postprocesses spectra through absorption by the IGM using the results of Meiksin [2006].

13.42.3. Madau (1995) IGM Attenuation

Selected with `stellarPopulationSpectraPostprocess<setName>Method=madau1995` this method postprocesses spectra through absorption by the IGM using the results of Madau [1995].

13.42.4. Lyman-continuum Suppression

Selected with `stellarPopulationSpectraPostprocess<setName>Method=lycSuppress` this method suppresses all emission in the Lyman continuum.

13.42.5. Recent Star Formation

Selected with `stellarPopulationSpectraPostprocess<setName>Method=recent` this method suppresses all emission from populations older than `[recentPopulationsTimeLimit]`.

13.42.6. Identity

Selected with `stellarPopulationSpectraPostprocess<setName>Method=identity` this method leaves the spectrum unchanged.

13.43. Stellar Astrophysics

Various properties related to stellar astrophysics are required by GALACTICUS. The following documents their implementation.
### 13.43. Basics

This subset of properties include recycled mass, metal yield and lifetime. The method used to compute such properties is specified by the `stellarAstrophysicsMethod` parameter.

**File**

Selected with `stellarAstrophysicsMethod=file` this method uses reads properties of individual stars of different initial mass and metallicity from an XML file and interpolates in them. The stars can be irregularly spaced in the plane of initial mass and metallicity. The XML file should have the following structure:

```xml
<stars>
  <star>
    <initialMass>0.6</initialMass>
    <lifetime>28.19</lifetime>
    <metallicity>0.0000</metallicity>
    <ejectedMass>7.65</ejectedMass>
    <metalYieldMass>0.44435954</metalYieldMass>
    <elementYieldMassFe>2.2017e-13</elementYieldMassFe>
    <source>Table 2 of Tumlinson, Shull &amp; Venkatesan (2003, ApJ, 584, 608)</source>
  </star>
  .
  .
  .
</stars>
```

Each star element must contain the `initialMass` (given in $M_\odot$) and `metallicity` tags. Other tags are optional. `lifetime` gives the lifetime of such a star (in Gyr), `ejectedMass` gives the total mass (in $M_\odot$) ejected by such a star during its lifetime, `metalYieldMass` gives the total mass of metals yielded by the star during its lifetime while `elementYieldMassFe` gives the mass of element Fe yielded by the star during its lifetime. The `source` and `url` tags are not used, but are strongly recommended to provide a reference to the origin of the stellar data.

### 13.43.2. Stellar Winds

Energy input to the ISM from stellar winds is used in calculations of feedback efficiency. The method used to compute stellar wind properties is specified by the `stellarWindsMethod` parameter.

**Leitherer et al. (1992)**

Selected with `stellarWindsMethod=Leitherer1992` this method uses the fitting formulae of Leitherer et al. [1992] to compute stellar wind energy input from the luminosity and effective temperature of a star.
13.43.3. Stellar Tracks

The method used to compute stellar tracks is specified by the `stellarTracksMethod` parameter.

**File**

Selected with `stellarTracksMethod=file` in this method luminosities and effective temperatures of stars are computed from a tabulated set of stellar tracks. The file containing the tracks to use is specified via the `stellarTracksFile` parameter. The file specified must be an HDF5 file with the following structure:

```plaintext
stellarTracksFile
  | +-> metallicity1
  |     | +-> metallicity
  |     | +-> mass1
  |     |     | +-> mass
  |     |     | +-> age
  |     |     | +-> luminosity
  |     |     | +-> effectiveTemperature
  |     +-> massN
  +-> metallicityN
```

Each `metallicityN` group tabulates tracks for a given metallicity (the value of which is stored in the `metallicity` dataset within each group), and may contain an arbitrary number of `massN` groups. Each `massN` group should contain a track for a star of some mass (the value of which is given in the `mass` dataset). Within each track three datasets specify the `age` (in Gyr), `luminosity` (in $L_\odot$) and `effectiveTemperature` (in Kelvin) along the track.

13.43.4. Supernovae Type Ia

Properties of Type Ia supernovae, including the cumulative number occurring and metal yield, are handled by the method selected using the `supernovaeIaMethod` parameter.

**Nagashima et al. (2005) Prescription**

Selected with `supernovaeIaMethod=Nagashima` this method uses the prescriptions from Nagashima et al. [2005] to compute the numbers and yields of Type Ia supernovae.

13.43.5. Population III Supernovae

Properties of Population III specific supernovae are handled by the method selected with the `supernovaePopIIIMethod` parameter.
Heger & Woosley (2002)
Selected with supernovaePopIIIMethod=Heger-Woosley2002 this method computes the energies of pair instability supernovae from the results of Heger and Woosley [2002].

13.43.6. Stellar Feedback
Aspects of stellar feedback are computed by the method selected with the stellarFeedbackMethod parameter.

Standard
Selected with stellarFeedbackMethod=standard, the method assumes that the cumulative energy input from a stellar population is equal to the total number of (Type II and Type Ia) supernovae multiplied by supernovaEnergy (specified in ergs) plus any Population III-specific supernovae energy plus the integrated energy input from stellar winds. The minimum mass of a star required to form a Type II supernova is specified (in $M_\odot$) via the initialMassForSupernovaeTypeII parameter.

13.44. Substructure and Merging
Substructures and merging of nodes/substructures is controlled by several algorithms which are described below:

13.44.1. Merging Timescales
The method used to compute merging timescales of substructures is specified by the satelliteMergingTimescalesMethod parameter.

Lacey & Cole (1993)
Selected with satelliteMergingTimescalesMethod=laceyCole1993, this method computes merging timescales using the dynamical friction calculation of Lacey and Cole [1993]. Timescales are multiplied by the value of the mergingTimescaleMultiplier input parameter.

Lacey & Cole (1993) + Tormen
Selected with satelliteMergingTimescalesMethod=laceyCole1993Tormen, this method computes merging timescales using the dynamical friction calculation of Lacey and Cole [1993] with a parameterization of orbital parameters designed to fit the results of Tormen [1997] as described by Cole et al. [2000]. Timescales are multiplied by the value of the mergingTimescaleMultiplier input parameter. Specifically, the merging time is taken to be:

$$\tau_{\text{merge}} = \frac{f_r \Phi \tau_{\text{dynamical}}}{2B(1)} \frac{M_{\text{host}}/M_{\text{satellite}}}{\ln(M_{\text{host}}/M_{\text{satellite}})}$$

(13.198)

where $f_r =$mergingTimescaleMultiplier, $\tau_{\text{dynamical}}$ is the dynamical time of the host halo and $B(x) = \text{erf}(x) - 2x \exp(x)/\sqrt{\Pi}$. The orbital factor $\Phi = e^{0.78 (R_c/R_{\text{vir}})^2}$ is drawn at random from a log-normal distribution with median $-0.14$ and dispersion $0.26$ as found by Cole et al. [2000].
13. Physical Implementations

**Jiang (2008)**

Selected with `satelliteMergingTimescalesMethod=jiang2008`, this method computes merging timescales using the dynamical friction calibration of Jiang et al. [2008]. Jiang et al. [2008] find that their fitting formula does not perfectly capture the results of N-body simulations, instead showing a scatter in the ratio $T_{fit}/T_{sim}$ where $T_{fit}$ is the merging time from their fitting formula and $T_{sim}$ is that measured from their N-body simulation. Furthermore, they show that the distribution of $T_{fit}/T_{sim}$ is well described by a log-normal with dispersion (in log($T_{fit}/T_{sim}$)) of $\sigma = 0.4$. Random perturbations can be applied to the merger times returned by this implementation by setting $\sigma = \text{[satelliteMergingJiang2008Scatter]} > 0$ which will cause the merger time to be drawn from a log-normal distribution of width $\sigma$ with median equal to $T_{fit}$.

**Boylan-Kolchin (2008)**

Selected with `satelliteMergingTimescalesMethod=boylanKolchin2008`, this method computes merging timescales using the dynamical friction calibration of Boylan-Kolchin et al. [2008].

**Wetzel & White (2010)**

Selected with `satelliteMergingTimescalesMethod=wetzelWhite2010`, this method computes merging timescales using the dynamical friction calibration of Wetzel and White [2010].

**Villalobos et al. (2013)**

Selected with `satelliteMergingTimescalesMethod=villalobos2013`, this method computes merging timescales using the modifier of Villalobos et al. [2013] as

$$
\tau_{\text{merge}} = (1 + z)^{\alpha} \tau'_{\text{merge}},
$$

where $\alpha = \text{[satelliteMergingTimescaleVillalobos2013Exponent]}$ and $\tau'_{\text{merge}}$ is the merging timescale computed by another satellite merging timescale method as specified via `satelliteMergingTimescaleVillalobos2013BaseMethod`.

**Preset**

Selected with `satelliteMergingTimescalesMethod=preset`, this method returns a precomputed merging timescale stored by the satellite component.

**Null**

Selected with `satelliteMergingTimescalesMethod=null`, this method returns zero merging timescale.

**Infinite**

Selected with `satelliteMergingTimescalesMethod=infinite`, this method returns an infinite merging timescale (technically, it returns $10^{30}$ Gyr which should be sufficiently close to infinity for practical purposes).

13.44.2. Virial Orbits

The algorithm to be used to determine orbital parameters of substructures when they first enter the virial radius of their host is specified via the `virialOrbitsMethod` parameter.
Benson (2005)

Selected with $\text{virialOrbitsMethod} = \text{Benson2005}$, this method selects orbital parameters randomly from the distribution given by Benson [2005].

Fixed

Selected with $\text{virialOrbitsMethod} = \text{fixed}$, this method sets all orbital parameters to fixed values, with $v_r = [\text{virialOrbitsFixedRadialVelocity}]V_{\text{virial}}$ and $v_\phi = [\text{virialOrbitsFixedTangentialVelocity}]V_{\text{virial}}$.

Wetzel (2010)

Selected with $\text{virialOrbitsMethod} = \text{Wetzel2010}$, this method selects orbital parameters randomly from the distribution given by Wetzel [2010], including the redshift and mass dependence of the distributions. Note that the parameter $R_1$ can be come negative (which is unphysical) for certain regimes of mass and redshift according to the fitting function for $R_1$ given by Wetzel [2010]. Therefore, we enforce $R_1 > 0.05$. Similarly, the parameter $C_1$ can become very large in some regimes which is probably an artifact of the fitting function used rather than physically meaningful (and which causes numerical difficulties in evaluating the distribution). We therefore prevent $C_1$ from exceeding 9.999999$^9$

13.44.3. Node Merging

The algorithm to be used to process nodes when they become substructures is specified by the $\text{nodeMergersMethod}$ parameter.

Single Level Hierarchy

Selected with $\text{nodeMergersMethod} = \text{singleLevelHierarchy}$, this method maintains a single level hierarchy of substructure, i.e. it tracks only substructures, not sub-substructures or deeper levels. When a node first becomes a satellite it is appended to the list of satellites associated with its host halo. If the node contains its own satellites they will be detached from the node and appended to the list of satellites of the new host (and assigned new merging times).

13.45. Supernovae Feedback Models

The supernovae feedback driven outflow rate is computed using the method specified by the $\text{starFormationFeedback[Disks|Spheroids]}$ parameter for disks and spheroids respectively.

13.45.1. Fixed

Selected with $\text{starFormationFeedbackDisksMethod} = \text{fixed}$, this method assumes an outflow rate of:

$$
\dot{M}_{\text{outflow}} = f_{\text{outflow}} \frac{\dot{E}}{E_{\text{canonical}}},
$$

(13.200)

where $f_{\text{outflow}} = [\text{diskOutflowFraction}]$ is the fraction of the star formation rate that goes into outflow, $\dot{E}$ is the rate of energy input from stellar populations and $E_{\text{canonical}}$ is the total energy input by a canonical stellar population normalized to $1 M_\odot$ after infinite time.

$^9$We use this value rather than 10 since the GSL $_2F_1$ hypergeometric function fails in some cases when $C_1 \geq 10$. 

303
13. Physical Implementations

### 13.45.2. Power Law

Selected with `starFormationFeedback[Disks|Spheroids]Method=powerLaw`, this method assumes an outflow rate of:

\[
\dot{M}_{\text{outflow}} = \left( \frac{\dot{V}_{\text{outflow}}}{V} \right)^{\alpha_{\text{outflow}}} \frac{\dot{E}}{E_{\text{canonical}}},
\]

where \(\dot{V}_{\text{outflow}}\)=[disk|spheroid]OutflowVelocity (in km/s) and \(\alpha_{\text{outflow}}\)=[disk|spheroid]OutflowVelocity are input parameters, \(V\) is the characteristic velocity of the component, \(\dot{E}\) is the rate of energy input from stellar populations and \(E_{\text{canonical}}\) is the total energy input by a canonical stellar population normalized to 1\(M_\odot\) after infinite time.

### 13.45.3. Creasey et al. (2012)

Selected with `starFormationFeedbackDisksMethod=Creasey2012`, this method computes the outflow rate using the model of Creasey et al. [2012]. Specifically,

\[
\dot{M}_{\text{outflow}} = \frac{\dot{E}_{\text{SN}}}{E_{\text{SN}}M_*} \int_0^\infty \beta_0 \Sigma_g^{\nu}(r)f_g(r)\dot{\Sigma}_\star(r)2\pi r dr,
\]

where \(\Sigma_g,1(r)\) is the surface density of gas in units of \(M_\odot \text{pc}^{-2}\), \(f_g(r)\) is the gas fraction, \(\dot{\Sigma}_\star(r)\) is the surface density of star formation rate, \(M_*\) is the total star formation rate in the disk, \(E_{\text{SN}}\) is the current energy input rate from supernovae, \(E_{\text{SN}}\) is the total energy input per unit mass from a stellar population after infinite time, \(\beta_0=[\text{starFormationFeedbackDisksCreasy2012Beta0}]\), \(\mu=[\text{starFormationFeedbackDisksCreasy2012Mu}]\), and \(\nu=[\text{starFormationFeedbackDisksCreasy2012Nu}]\).

### 13.46. Supernovae Expulsive Feedback Models

The expulsive supernovae feedback driven outflow rate is computed using the method specified by the `starFormationExpulsiveFeedback[Disks|Spheroids]Method` for disks and spheroids respectively.

#### 13.46.1. Null

Selected with `starFormationExpulsiveFeedback[Disks|Spheroids]Method=null`, this method assumes a zero outflow rate.

#### 13.46.2. Superwind

Selected with `starFormationExpulsiveFeedback[Disks|Spheroids]Method=superwind`, this method assumes an outflow rate of:

\[
\dot{M}_{\text{outflow}} = \beta_{\text{superwind}} \frac{\dot{E}}{E_{\text{canonical}}} \begin{cases} 
(V_{\text{superwind}}/V)^2 & \text{if } V > V_{\text{superwind}} \\
1 & \text{otherwise}
\end{cases}
\]

where \(V_{\text{superwind}}=[\text{disk|spheroid}](\text{SuperwindVelocity} \text{ (in km/s)) and } \beta_{\text{superwind}}=[\text{disk|spheroid}](\text{SuperwindMassLoading})\) are input parameters, \(V\) is the characteristic velocity of the component, \(\dot{E}\) is the rate of energy input from stellar populations and \(E_{\text{canonical}}\) is the total energy input by a canonical stellar population normalized to 1\(M_\odot\) after infinite time.
13.47. Supermassive Black Hole Binaries: Initial Separation

The method to be used for computing the initial separation of black hole binaries is specified by the blackHoleBinaryInitialRadiiMethod parameter.

13.47.1. Spheroid Radius Fraction

Selected with blackHoleBinaryInitialRadiiMethod=spheroidRadiusFraction, this method assumes that the initial separation of the binary is equal to a fixed fraction [blackHoleInitialRadiusSpheroidRadiusRatio] of the larger of the spheroid scale radii of the two merging galaxies.


Selected with blackHoleBinaryInitialRadiiMethod=Volonteri2003, this method assumes that the initial separation follows the relationship described in Volonteri et al. [2003]

\[ r_{\text{initial}} = \frac{G(M_{*1} + M_{*2})}{2\sigma_{DM}^2} \]  \hspace{1cm} (13.204)

where \( M_{*1} \) and \( M_{*2} \) are the masses of the black holes and \( \sigma_{DM} \) is the velocity dispersion of the dark matter, which we assume to equal the virial velocity of the dark matter halo.

13.47.3. Tidal Radius

Selected with blackHoleBinaryInitialRadiiMethod=tidalRadius, this method assumes an initial separation that corresponds to the distance at which the satellite galaxy is tidally stripped to its half-mass radius, thus only leaving the central massive black hole. Specifically, the initial radius is given by:

\[ \frac{M_{\text{sat}}}{2r_{\text{sat},1/2}^3} = -\frac{d}{dr} \frac{M_{\text{host}}(r_{\text{initial}})}{r_{\text{initial}}^2} \]  \hspace{1cm} (13.205)

Where \( M_{\text{sat}} \) is the mass of the satellite galaxy, \( r_{\text{sat},1/2} \) is its half mass radius, \( M_{\text{host}}(r) \) is the mass of the host galaxy within radius \( r \) and \( r_{\text{initial}} \) is the initial radius.

13.48. Supermassive Black Hole Binaries: Separation Growth Rate

The method to be used for computing the separation growth rate of black hole binaries is specified by the blackHoleBinarySeparationGrowthRateMethod parameter.

13.48.1. Null

Selected by default and with blackHoleBinarySeparationGrowthRateMethod=null, this method assumes that the initial separation of the binaries is final.

13.48.2. Standard

Selected with blackHoleBinarySeparationGrowthRateMethod=Standard, this method computes the separation growth rate of the binaries following a modified version of Volonteri et al. [2003] which include
terms for dynamical friction, hardening due to scattering of stars and gravitational wave emission.

\[ \dot{a} = \min \left( - \frac{G \rho_* a^2 H}{\sigma}, + \frac{2 \dot{v}_{DF} a}{v_c} \right) - \frac{256 G^3 M_{*,1} M_{*,2} (M_{*,1} + M_{*,2})}{5 c^5 a^3} \]  

(13.206)

where \( a \) is the black hole binary separation, \( H \) is a dimensionless hardening parameter \( H \approx 15 \) in the limit of a very hard, equal mass binary, \( \rho_* \) is the density of stars, \( \dot{v}_{DF} \) is the acceleration (negative) due to dynamical friction, \( v_c \) is the circular velocity, \( \sigma \) is the velocity dispersion of stars. Here the first factor represents hardening due to strong scattering of stars, the second results from dynamical friction with distant stars, gas and dark matter and the last results from the emission of gravitational waves Peters [1964].

The acceleration due to dynamical friction is computed using Chandrasekhar’s formula:

\[ \dot{v}_{DF} = - \frac{2 \pi G^2 M_*}{V_C^2} \sum_i \rho_i \log(1 + \Lambda_i^2) \left[ \text{erf}(X_i) - \left\{ \frac{2 X_i}{\sqrt{\pi}} \exp \left( -X_i^2 \right) \right\} \right], \]  

(13.207)

where the sum is taken over the spheroid (gaseous plus stellar mass) and dark matter halo components\(^{10}\).

Here,

\[ \Lambda_i = \frac{a \sigma^2}{G(M_{*,1} + M_{*,2})}, \]  

(13.208)

is the Coulomb logarithm and

\[ X_i = V_c / \sqrt{2} \sigma. \]  

(13.209)

In all of the above equations, the velocity dispersion \( \sigma_i \) is computed from the spherical Jeans equation assuming an isotropic velocity dispersion if \([\text{blackHoleBinariesComputeVelocityDispersion}]=\text{true}\). Otherwise, \( \sigma_i \) is set to the halo virial velocity for dark matter and to the spheroid characteristic velocity for the spheroid.

In calculating the rate of hardening due to scattering of stars, the stellar density is reduced by a factor \([\text{Volonteri et al., 2003}]\)

\[ f_\rho = \min \left\{ \left( \frac{4 a \sigma^2_{\text{spheraoid}}}{3 G(M_{*,1} + M_{*,2})} \log \left( \frac{G M_{*,2}}{4 a \sigma^2_{\text{spheraoid}} a} \right) \right)^2, 1 \right\}, \]  

(13.210)

if \([\text{stellarDensityChangeBinaryMotion}]=\text{true}\) to account for the ejection of stars from the loss cone.

### 13.49. Supermassive Black Holes Binaries: Recoil Velocity

The method to be used for computing the recoil velocity due to gravitational waves ejection during a binary merger specified by the \texttt{blackHoleBinaryRecoilVelocityMethod} parameter.

#### 13.49.1. Null

Selected by default and with \texttt{blackHoleBinaryRecoilVelocityMethod=null}, this method assumes that there is no recoil velocity.

---

\(^{10}\)The disk is ignored as the black hole is assumed to be orbiting in a circular orbit in the disk.
13.49.2. Campanelli et al. (2007)

Selected with `blackHoleBinaryRecoilVelocityMethod=Campanelli2007`, this method computes the recoil velocity during a black hole binary merger due to the emission of gravitational waves, following the formulae derived in Campanelli et al. [2007]

\[ V_{\text{recoil}} = V_m \epsilon_1 + V_\perp (\cos \xi \epsilon_1 + \sin \xi \epsilon_2) + V_\parallel \epsilon_2 \]  \hspace{1cm} (13.211)

with:

\[ V_m = A q^2 (1-q) \frac{1}{(1+q)^5} (1 + B \frac{q}{(1+q)^2}) \] \hspace{1cm} (13.212)

\[ V_\perp = H \frac{q^2}{(1+q)^5} (\alpha_{2 \parallel} - q \alpha_{1 \parallel}) \] \hspace{1cm} (13.213)

\[ V_\parallel = K \cos(\theta - \theta_0) \frac{q^2}{(1+q)^5} (\alpha_{2 \perp} - q \alpha_{1 \perp}) \] \hspace{1cm} (13.214)

where \( \theta \) is defined as the angle between the inplane component of \( \Delta \) and the infall direction at merger. \( q \) is the mass ratio of the black holes as \( q = M_{\bullet,1}/M_{\bullet,2} \) and \( \alpha_i = S_i/M_{\bullet,i} \) depends of the spin and mass of the black hole \( \xi \) measures the angle between the unequal mass and the spin contribution to the recoil velocity in the orbital plane. \( \epsilon_1, \epsilon_2 \) are orthogonal unit vectors in the orbital plane. Our method assumes the spin of the second black hole is randomly generated, while that of the first is aligned with the angular momentum of the system. The constants used are retrieved from the articles by: Koppitz et al. [2007] for \( H = (7.3 \pm 0.3) \times 10^3 \) km/s, González et al. [2007b] for \( A = 1.2 \times 10^4 \) km/s, González et al. [2007a] for \( K \cos(\delta \theta) = (6, -5.3) \times 10^4 \) km/s and \( K = (6.0 \pm 0.1) \times 10^4 \) km/s.

13.50. Supermassive Black Hole Binaries: Mergers

The method to be used for computing the effects of binary mergers of supermassive black holes is specified by the `blackHoleBinaryMergersMethod` parameter.

13.50.1. Rezzolla et al. (2008)

Selected with `blackHoleBinaryMergersMethod=Rezzolla2008`, this method uses the fitting function of Rezzolla et al. [2008] to compute the spin of the black hole resulting from a binary merger. The mass of the resulting black hole is assumed to equal the sum of the mass of the initial black holes (i.e. there is negligible energy loss through gravitational waves).

13.51. Tidal Induced Mass Loss Rates in Disks/Spheroids

The methods for computing tidal induced rates of mass loss in disks/spheroids are selected via the `tidalStrippingMassLossRate(Disks|Spheroids)Method` parameter.

13.51.1. Simple

Selected with `tidalStrippingMassLossRate(Disks|Spheroids)Method=simple` this method computes the mass loss rate to be:

\[ \dot{M} = -\alpha M/\tau_{(disk|spheroid)}, \]  \hspace{1cm} (13.215)

where

\[ \alpha = \min(F_{\text{tidal}}/F_{\text{gravity}}, R_{\text{maximum}}), \] \hspace{1cm} (13.216)
13. Physical Implementations

\[ F_{\text{tidal}} = F_{\text{tidal}}r_{1/2}, \ F_{\text{tidal}} \] is the tidal field from the host halo (see §13.35),
\[ F_{\text{gravity}} = V_{1/2}^2(r_{1/2})/r_{1/2} \quad (13.217) \]
is the gravitational restoring force in the disk/spheroid at the half-mass radius, \( r_{1/2} \), and \( R_{\text{maximum}} = [\text{tidalStrippingMassLossRate} \ (\text{Disks}|\text{Spheroids}) \ \text{Method}] \).

13.51.2. Null

Selected with \text{tidalStrippingMassLossRate(Disks|Spheroids)Method=\textbf{null}}\ this method assumes zero mass loss rate always.

13.52. Survey Geometry

The method to be used for computing the geometry of surveys for data analysis is specified by the \text{surveyGeometryMethod} parameter.

13.52.1. Li & White (2009)

Selected with \text{surveyGeometryMethod=\textbf{Li-White-2009-SDSS}}, this method describes the survey geometry of Li and White [2009].

For the angular mask, we make use of the catalog of random points within the survey footprint provided by the NYU-VAGC\textsuperscript{11} (Blanton et al. 2005; see also Adelman-McCarthy et al. 2008, Padmanabhan et al. 2008). Li and White [2009] consider only the main, contiguous region and so we keep only those points which satisfy \( \text{RA} > 100^\circ \), \( \text{RA} < 300^\circ \), and \( \text{RA} < 247^\circ \) or \( \delta < 51^\circ \). When the survey window function is needed, these points are used to determine which elements of a 3D grid fall within the window function.

To estimate the depth of the Li and White [2009] sample as a function of galaxy stellar mass we make use of semi-analytic models in the Millennium Database. Specifically, we use the semi-analytic model (SAM) of De Lucia and Blaizot (2007; specifically the millimil..DeLucia2006a and millimil..DeLucia2006a..sdss2mass tables in the Millennium Database). For each snapshot in the database, we extract the stellar masses and observed-frame SDSS r-band absolute magnitudes (including dust extinction), and determine the median absolute magnitude as a function of stellar mass. Using the limiting apparent magnitude of the Li and White [2009] sample, \( r = 17.6 \), we infer the corresponding absolute magnitude at each redshift and, using our derived absolute magnitude–stellar mass relation, infer the corresponding stellar mass.

The end result of this procedure is the limiting stellar mass as a function of redshift, accounting for k-corrections, evolution, and the effects of dust. Figure 13.1 shows the resulting relation between stellar mass and the maximum redshift at which such a galaxy would be included in the sample. Points indicate measurements from the SAM, while the line shows a polynomial fit:

\[ z(M_*) = -5.950 + 2.638m - 0.4211m^2 + 2.852 \times 10^{-2}m^3 - 6.783 \times 10^{-4}m^4, \quad (13.218) \]

where \( m = \log_{10}(M_*/M_\odot) \). We use this polynomial fit to determine the depth of the sample as a function of stellar mass. We adopt a solid angle of \( 2.1901993 \) sr [Percival et al., 2007] for the sample.

13.52.2. Martin et al. (2010)

Selected with \text{surveyGeometryMethod=\textbf{Martin-2010-ALFALFA}}, this method describes the survey geometry of Martin et al. [2010].

\textsuperscript{11}Specifically, \url{http://sdss.physics.nyu.edu/lss/dr72/random/lss_random-0.dr72.dat}.
Figure 13.1.: The maximum redshift at which a galaxy of given stellar mass can be detected in the sample of Li and White [2009]. Points show the results obtained using the De Lucia and Blaizot [2007] model from the Millennium Database, while the lines shows a polynomial fit to these results (given in eqn. 13.218).

For the angular mask we use the three disjoint regions defined by $07^h30^m < R.A. < 16^h30^m$, $+04^\circ < \text{decl.} < +16^\circ$, and $+24^\circ < \text{decl.} < +28^\circ$ and $22^h < R.A. < 03^h$, $+14^\circ < \text{decl.} < +16^\circ$, and $+24^\circ < \text{decl.} < +32^\circ$ corresponding to the sample of Martin et al. [2010]. When the survey window function is needed we generate randomly distributed points within this angular mask and out to the survey depth. These points are used to determine which elements of a 3D grid fall within the window function.

To estimate the depth of the Martin et al. [2010] sample as a function of galaxy HI mass we first infer the median line width corresponding to that mass. To do so, we have fit the median line width-mass relation from the $\alpha.40$ sample with power-law function as shown in Fig. 13.2. We find that the median line width can be approximated by

$$\log_{10}(W_{50}/\text{km s}^{-1}) = c_0 + c_1 \log_{10}(M_{\text{HI}}/M_{\odot}),$$

with $c_0 = -0.770$ and $c_1 = 0.315$. Given the line width, the corresponding integrated flux limit, $S_{\text{int}}$, for a signal-to-noise of 6.5 is inferred using equation (A1) of Haynes et al. [2011]. Finally, this integrated flux limit is converted to maximum distance at which the source could be detected using the expression given in the text of section 2.2 of Martin et al. [2010]:

$$M_{\text{HI}} = 2.356 \times 10^5 \left(\frac{D}{\text{Mpc}}\right)^2 \left(\frac{S_{\text{int}}}{\text{Jy km s}^{-1}}\right).$$
Figure 13.2.: HI line width vs. HI mass as measured from the $\alpha.40$ survey of Martin et al. [2010]. Red points with error bars show individual measurements, while the larger circles indicate the running median of these data. The green line is a power-law fit to the running median as described in eqn. (13.219).
14. Additional Output Quantities

14.1. Black Hole Accretion

Properties associated with accretion onto supermassive black holes can be output by setting blackHoleOutputAccretion=true. Currently, two additional properties are output for each node when this option is selected:

- blackHoleAccretionRate: The rate at which the supermassive black hole is accreting mass in \( M_\odot \text{ Gyr}^{-1} \);
- blackHoleJetPower: The power being emitted into jets by the black hole/accretion disk system in \( M_\odot \text{ km}^2 \text{ s}^{-2} \text{ Gyr}^{-1} \).

14.2. Cooling Data

Properties associated with cooling in hot halos can be output by setting hotHaloOutputCooling=true. Currently, two additional properties are output for each node when this option is selected:

- hotHaloCoolingRate: The rate at which gas is cooling from the halo (assuming no sources of heating) in \( M_\odot \text{ Gyr}^{-1} \);
- hotHaloCoolingRadius: The characteristic cooling radius in the halo in Mpc.

14.3. Density Contrast Data

Properties of nodes at density contrasts other than the virial density can be output by setting [outputDensityContrastData]=true. When selected, this output option requires that a list of density contrasts, \( \Delta \) (defined in units of the mean density of the Universe), be given in the [outputDensityContrastValues] input parameter. For each specified density contrast, two properties are output for each node: nodeRadius\( \Delta \) and nodeMass\( \Delta \) which give the radius enclosing a mean density contrast of \( \Delta \) and the mass enclosed within that radius. The parameter [outputDensityContrastDataDarkOnly] controls whether density contrasts are measured for total mass (false) or dark matter mass only (true). In the latter case, density contrasts are defined relative to the mean dark matter density of the Universe.

14.4. Descendent Node Index

By setting [outputDescendentIndices]=true the index of the node containing the galaxy to which each current galaxy will belong at the next output time (i.e. the forward descendent) will be written to the output file. To clarify, this will be the index of the node into which the galaxy descends, or the index of a node with which it merges prior to the next output time (and if that node merges with another, the index will be of that node and so on).

Note that, to operate correctly, information about which node a given node may merge with (and when this merger will happen) must be available. This is typically available in merger trees read from file providing [treeNodeMethodSatelliteOrbit] and [mergerTreeReadPresetMergerTimes] are both set to true. When using randomly assigned satellite orbits and merger times, information on when merging
occurs does not exist until a node becomes a satellite. Thus, if the node becomes a satellite after the current output, but before the next output, there is no way to know which node it will belong to at the next output (in such cases, the fallback assumption is no merging).

### 14.5. Half-Light Radii Data

Half-light radii and masses enclosed within them can be output by setting \([\text{outputHalfLightData}] = \text{true}\). When selected the half-light radius in each specified luminosity band is output as \([\text{halfLightRadius}\{\text{luminosityID}\}]\) (in Mpc), where \{luminosityID\} is the usual luminosity identifier suffix, and the total (dark + baryonic) mass within that radius is output as \([\text{halfLightMass}\{\text{luminosityID}\}]\) (in \(M_\odot\)).

### 14.6. Half-Mass Radii

Half-mass radii can be output by setting \([\text{outputHalfMassData}] = \text{true}\). When selected the half-mass radius is output as \([\text{halfMassRadius}]\) (in Mpc).

### 14.7. Halo Model Quantities

The following quantities related to galaxy clustering are output if \([\text{outputHaloModelData}] = \text{true}\):

- **nodeBias** The large scale, linear theory bias for each node. For satellite nodes, this corresponds to the bias of their host halo;
- **isolatedHostIndex** The index of the isolated node in which this node lives. This is identical to \(\text{nodeIndex}\) for non-satellite nodes.

In addition to these quantities output for each node, setting \([\text{outputHaloModelData}] = \text{true}\) causes the creation of a \(\text{haloModel}\) group in the \text{GALACTICUS} output file. This group contains the following:

- **wavenumber** A dataset giving the wavenumbers (in units of \(\text{Mpc}^{-1}\)) at which all output power spectra are tabulated. The minimum and maximum wavenumbers to tabulate are determined by the \([\text{haloModelWavenumberMinimum}]\) and \([\text{haloModelWavenumberMaximum}]\) parameters respectively, while the number of points to tabulate in each decade of wavenumber is determined by the \([\text{haloModelWavenumberPointsPerDecade}]\) parameter.
- **powerSpectrum** A dataset giving the linear theory power spectrum (in units of \(\text{Mpc}^3\) normalized to \(z = 0\) at each wavenumber specified in the \text{wavenumber} dataset.

\([\text{Output}\{i\}/\text{mergerTree}\{j\}/\text{fourierProfile}\{k\}]\) A dataset giving the Fourier transform of the dark matter halo density profile (dimensionless and normalized to unity at small wavenumber) for the node with index \(k\) in merger tree with index \(j\) at output number \(i\). Profiles are written only for nodes which are isolated, and are tabulated at the wavenumbers given in the \text{wavenumber} group. Note that wavenumbers are assumed to be comoving.

Finally, each numbered output group is given two additional attributes, **linearGrowthFactor** and **linearGrowthFactorLogDerivative** which give the growth factor, \(D\), and its logarithmic derivative, \(d\ln D/d\ln a\) at the output time.

The information output can be used to construct galaxy power spectra and correlation functions (see §5.2.2 for example).
14.8. Lightcone Coordinates

The position (and velocity and redshift) of a galaxy within a lightcone will automatically be output if the lightcone output filter is active (see §17.4.3). In such cases, the following properties will be output for all galaxies:

- **lightconePositionX**: Position of the galaxy (in comoving Mpc) along the radial direction of the lightcone;
- **lightconePositionY**: Position of the galaxy (in comoving Mpc) along the 1st angular direction of the lightcone;
- **lightconePositionZ**: Position of the galaxy (in comoving Mpc) along the 2nd angular direction of the lightcone;
- **lightconeVelocityX**: Velocity of the galaxy (in km/s) along the radial direction of the lightcone;
- **lightconeVelocityY**: Velocity of the galaxy (in km/s) along the 1st angular direction of the lightcone;
- **lightconeVelocityZ**: Velocity of the galaxy (in km/s) along the 2nd angular direction of the lightcone;
- **lightconeRedshift**: Redshift of the galaxy in the lightcone.

**angularWeight** The mean number density of this galaxy per unit area on the sky (in degrees^{-2}). For a “square” geometry, i.e. defined such that a point (x, y, z) is in the survey angular mask if |atan2(y, x)| < ψ/2 and |atan2(z, x)| < ψ/2 where atan2() is the quadrant-aware inverse tangent function, and ψ is the angular size of the field, we compute the solid angle of the lightcone as follows. Define a spherical coordinate system (θ, φ) with the pole (θ = 0) aligned with the x-axis. The solid angle of the field is then

\[
\Omega = 2\pi \int_0^{\psi/2} \sin \theta d\theta + 8 \int_{\psi/2}^{\tan^{-1}(\sqrt{2}\tan(\psi/2))} d\theta \sin \theta \int_{\cos^{-1}(\tan(\psi/2)/\tan \theta)}^{\pi/4} d\phi, \tag{14.1}
\]

which is

\[
\Omega = 2\pi [1 - \cos(\psi/2)] + 8 \int_{\psi/2}^{\tan^{-1}(\sqrt{2}\tan(\psi/2))} d\theta \sin \theta \left[ \frac{\pi}{4} - \cos^{-1}\left(\frac{\tan(\psi/2)}{\tan \theta}\right) \right], \tag{14.2}
\]

or

\[
\Omega = 2\pi [1 - \cos(\tan^{-1}(\sqrt{2}\tan(\psi/2)))] - 8 \int_{\psi/2}^{\tan^{-1}(\sqrt{2}\tan(\psi/2))} d\theta \sin \theta \cos^{-1}\left(\frac{\tan(\psi/2)}{\tan \theta}\right), \tag{14.3}
\]

The final integral can be evaluated (using Mathematica for example) to give

\[
\begin{align*}
\Omega &= 2\pi [3 - \cos(\tan^{-1}(\sqrt{2}\tan(\psi/2)))] - 8 \sin(x) \left( \sqrt{(a^2 + 1) \cos(2x) + a^2 - 1} \log\left((a\sqrt{2a^2 \cos^2(2x) + \cos(2x)} - 1 + 2a) - \log(\cos(2x) - 1)\right)csc^2(x) \left(-(a^2 + 1) \cos(2x) + a^2 - 1\right) - \cot(x)((a^2 + 1) \cos(2x) + a^2 - 1) \cos^{-1}(a \cot(x))) / ((a^2 + 1) \cos(2x) + a^2 - 1), \tag{14.4}
\end{align*}
\]

where \(a = \tan(\psi/2)\) and \(x = \tan^{-1}(\sqrt{2}\tan(\psi/2))\).

If active, the geometry of the lightcone must be specified in an XML file, the name of which must be specified using the [filterLightconeGeometryFileName] input parameter. The XML file should have the following structure:

\[1\] Note that this will not, in general, be precisely the same as the redshift corresponding to the output time.
14. Additional Output Quantities

<geometry>
  <boxLength>500</boxLength>
</geometry>

<cosmology>
  <parameter>
    <name>omega0</name>
    <value>0.25</value>
  </parameter>
  <parameter>
    <name>lambda0</name>
    <value>0.75</value>
  </parameter>
  <parameter>
    <name>H0</name>
    <value>70</value>
  </parameter>
</cosmology>

<fieldOfView>
  <geometry>square</geometry>
  <length>0.0174532927777778</length>
</fieldOfView>

<maximumDistance>6136.78369140625</maximumDistance>

<origin>
  <coordinate>50</coordinate>
  <coordinate>50</coordinate>
  <coordinate>50</coordinate>
</origin>

<outputs>
  <maximumDistance>9606.4736328125</maximumDistance>
  <maximumDistance>9470.22194965342</maximumDistance>
  ...
  <minimumDistance>9470.22194965342</minimumDistance>
  <minimumDistance>9160.59527164611</minimumDistance>
  ...
  <redshift>127</redshift>
  <redshift>79.997894</redshift>
  ...
</outputs>

<unitVector1>
  <coordinate>0.307692307692308</coordinate>
  <coordinate>0.230769230769231</coordinate>
  <coordinate>0.923076923076923</coordinate>
</unitVector1>
14.9. Main Branch Evolution

The boxLength element should give the length of the simulation box (the box will be replicated to span the volume covered by the lightcone), while the maximumDistance element should specify the largest distance in the lightcone to be considered. The fieldOfView element must specify the geometry of the field of view. Currently, the only allowed value for the geometry of the fieldOfView element is square, in which case the length element of fieldOfView should give the length of the side of the square field of view in radians. The origin element must contain the x, y, z coordinates of the origin of the lightcone within the simulation box, while the unitVectorX elements must give unit vectors which point along the lightcone (for X = 1) and in the two directions perpendicular to the lightcone (for X = 2 and 3). The outputs element contains lists of properties corresponding to the outputs to be used for building the lightcone. The redshift subelements must list the redshifts of available outputs (in order of decreasing redshift) while the minimumDistance and maximumDistance elements must give the minimum and maximum comoving distance that should be considered to be associated with each given output redshift. The units/length element must specify the units used for lengths, with unitsInSI giving the length unit in SI units and hubbleExponent giving the exponent of h that appears in the length unit.

Finally, the optional cosmology element may give cosmological parameters relevant to this lightcone in standard GALACTICUS format.

14.9. Main Branch Evolution

The evolution of main branch galaxies can be recorded by setting \[\text{timestepRecordEvolution} = \text{true}\]. When set, the evolution of each main branch galaxy will be recorded at a set of \[\text{timestepRecordEvolutionSteps}\] timesteps spaced logarithmically in cosmic time between \[\text{timestepRecordEvolutionBegin}\] and \[\text{timestepRecordEvolutionEnd}\].

This recorded evolution will be written to the group mainProgenitorEvolution in the GALACTICUS output file. Within that group two datasets, time and expansionFactor, give the times and expansion factors at which evolution was recorded. Then for each merger tree two datasets, stellarMass\(<N>\) and totalMass\(<N>\) (where \(<N>\) is the merger tree index), give the stellar and total baryonic mass of the main branch progenitor at each timestep.
14. Additional Output Quantities

14.10. Main Branch Status

The status of each node with respect to the main branch of its merger tree can be output by setting \([\text{outputMainBranchStatus}] = \text{true}\). When set, the status will be output as \(\text{nodeIsOnMainBranch}\), with a value of 1 indicating that the node is a primary progenitor of the final halo (i.e. is on the main branch of the tree) and a value of 0 indicating that it is not.

14.11. Mass Profile Data

Masses enclosed within specific radii can be output by setting \([\text{outputMassProfileData}] = \text{true}\). When selected, this output option requires that a list of radii, \(r\) (in Mpc), be given in the \([\text{outputMassProfileRadii}]\) input parameter. For each specified radius, the total (dark + baryonic) mass will be output as \(\text{massProfile}r\).

14.12. Merger Tree Links and Node Isolation

The following properties are output to permit the merger tree structure to be recovered:

- **nodeIndex**: A unique (within a tree) integer index identifying the node;
- **childIndex**: The index of this node’s primary child node (or \(-1\) if it has no child);
- **parentIndex**: The index of this node’s parent node (or \(-1\) if it has no parent);
- **siblingIndex**: The index of this node’s sibling node (or \(-1\) if it has no sibling);
- **satelliteIndex**: The index of this node’s first satellite node (or \(-1\) if it has no satellites);
- **nodeIsIsolated**: Will be 0 for a node which is a subhalo inside some other node (i.e. a satellite galaxy) or 1 for a node that is an isolated halo (i.e. a central galaxy).

The **nodeIndex** property corresponds by default to the index of the node in the original merger tree. This means that as a galaxy evolves through the tree and, in particular, gets promoted into a new halo the index associated with a galaxy will change. This is useful to identify where the galaxy resides in the original (unevolved) tree structure, but does not allow galaxies to be traced from one output to the next using their **nodeIndex** value. By setting \([\text{nodePromotionIndexShift}] = \text{true}\) this behavior can be changed such that the value of **nodeIndex** will reflect the index of the earliest progenitor node along the main branch of the current node. As such, this index will remain the same for a given galaxy during its evolution. These two alternative algorithms for propagating node indices are illustrated in Figure 14.1.

14.13. Merger Tree Data for Rendering

Data on the structure of a merger tree and its halos useful for rendering the tree as a 3-D structure can be output using the **Merger_Trees_Render** module. Calling **Merger_Trees_Render_Dump** with a tree as the only argument will cause the tree structure will be dumped to a file named \(\text{render}_{\langle \text{treeIndex} \rangle}_ {-\langle \text{outputIndex} \rangle}.\text{hdf5}\) where \(\langle \text{treeIndex} \rangle\) is the index of the tree and \(\langle \text{outputIndex} \rangle\) is an incremental counter that tracks the number of outputs for this tree. The output is a simple HDF5 file containing the following datasets:

- **nodeIndex**: Index of the node;
- **parentIndex**: Index of the parent node;
Figure 14.1.: Illustration of options for the propagation of node indices during node promotion events. Two identical trees (top row) are evolved with $\text{[nodePromotionIndexShift]}=\text{false}$ (left column) and $\text{[nodePromotionIndexShift]}=\text{true}$ (right column). The middle and lower rows indicate the resulting node indices after two stages of tree evolution.
14. Additional Output Quantities

**childIndex** Index of the child node;

**time** Time of the node;

**expansionFactor** Corresponding expansion factor;

**radiusVirial** Virial radius of the node;

**position** \((x, y, z)\) position of the node.

### 14.14. Merger Tree Structure

The structure of each merger tree can optionally be dumped to a file suitable for post-processing with **DOT** after every step of evolution. To request this output set \([\text{mergerTreesDumpStructure}]=\text{true}\). After each evolution step the tree structure will be dumped to a file named \(\text{mergerTreeDump:}\langle\text{treeIndex}\rangle:\langle\text{outputIndex}\rangle\.gv\) where \(\langle\text{treeIndex}\rangle\) is the index of the tree and \(\langle\text{outputIndex}\rangle\) is an incremental counter that tracks the number of outputs for this tree. These files can be processed with **DOT** to produce a diagram of the tree structure. The node currently being evolved will be highlighted in green. This output option makes use of the **Merger_Trees_Dump** module to create the outputs.

### 14.15. Most Massive Progenitor

Setting \([\text{outputMostMassiveProgenitor}]=\text{true}\) causes the property \(\text{isMostMassiveProgenitor}\) to be output. This property will be 1 for the most massive progenitor node in a tree at each output time and 0 for all other nodes.

### 14.16. Rotation Curve

Setting \([\text{outputRotationCurveData}]=\text{true}\) causes the rotation curve of the node to be output at specified radii. The radii and types of rotation curve to output is specified by the \(\text{outputRotationCurveRadii}\) parameter. This parameter’s value can contain multiple entries. Each entry is of the form:

\[
\text{radiusType:componentType:massType:loading?:radius}
\]

The elements of this colon-separated specifier determine the radius at which the rotation curve is computed, which components/mass types should be counted, and whether baryonic loading of the halo should be accounted for. The elements have the following meaning:

- **radius** the numerical value of the radius at which to compute the rotation curve (with units specified by the **radiusType** element);

- **radiusType** specifies the units of the **radius** element—valid options are **diskRadius**, **diskHalfMassRadius**, **spheroidRadius**, **spheroidHalfMassRadius**, **darkMatterScaleRadius**, **virialRadius**, just **radius** (which implies radii are given in units of Mpc), **galacticMassFraction{<fraction>}**, or **galacticLightFraction{<fraction>}{<luminosity>}**, where the final two form specify a radius containing a fixed **fraction** of the galactic mass or light respectively (for the case of galactic light, **luminosity** specifies the band, e.g. **SDSS\_r:rest:z0.0000**);

- **componentType** specifies which components of the node should be counted—allowed values are **all**, **disk**, **spheroid**, **hotHalo**, **darkHalo**, and **blackHole**;
massType specifies which types of mass should be counted—allowed values are all, dark, baryonic, galactic, gaseous, stellar, and blackHole;

loading? option should be either loaded or unloaded, and specifies whether the effect of baryonic loading (i.e. adiabatic contraction) should be included in the calculation of the rotation curve.

### 14.17. Satellite Orbital Pericenter

Setting [outputSatellitePericenterData] to true will cause the pericentric values of the radius and velocity of each satellite node’s orbit to be output as satellitePericenterRadius and satellitePericenterVelocity respectively.

### 14.18. Star Formation Rates

By default the star formation rate in each galaxy is not output. However, setting [diskOutputStarFormationRate] to true will cause the current star formation rate in the disk of each galaxy to be output as diskStarFormationRate (in units of $M_\odot$/Gyr). The [spheroidOutputStarFormationRate] has the same effect for the spheroid component.

### 14.19. Velocity Dispersion

Setting [outputVelocityDispersionData]=true causes the velocity dispersion of the node to be output at specified radii, for specified components. The radii, components and type of velocity dispersion to output are specified by the outputVelocityDispersionRadii parameter. This parameter’s value can contain multiple entries. Each entry is of the form:

```
radiusType:componentType:massType:loading?:direction:radius
```

The elements of this colon-separated specifier determine the radius at which the velocity dispersion is computed, which component/mass type the velocity dispersion should be computed for, whether baryonic loading of the halo should be accounted for, and for which direction the velocity dispersion should be computed. The elements have the following meaning:

- **radius** the numerical value of the radius at which to compute the velocity dispersion (with units specified by the radiusType element);
- **radiusType** specifies the units of the radius element—valid options are diskRadius, diskHalfMassRadius, spheroidRadius, spheroidHalfMassRadius, darkMatterScaleRadius, virialRadius, just radius (which implies radii are given in units of Mpc), galacticMassFraction{<fraction>}, or galacticLightFraction{<fraction>{<luminosity>}, where the final two form specify a radius containing a fixed <fraction> of the galactic mass or light respectively (for the case of galactic light, <luminosity> specifies the band, e.g. SDSS_r:rest:z0.0000);
- **componentType** specifies which component of the node the velocity dispersion should be computed for—allowed values are all, disk, spheroid, hotHalo, darkHalo, and blackHole²;
- **massType** specifies which type of mass the velocity dispersion should be computed for—allowed values are all, dark, baryonic, galactic, gaseous, stellar, and blackHole;

²Note that attempting to compute the velocity dispersion for a black hole or disk for example won’t make any sense.
loading? option should be either loaded or unloaded, and specifies whether the effect of baryonic loading (i.e. adiabatic contraction) should be included in the calculation of the velocity dispersion.

direction should be one of radial (computes the radial component of velocity dispersion), lineOfSight{<luminosity>} (computes the line-of-sight velocity dispersion), lineOfSightInteriorAverage{<luminosity>} (computes the line-of-sight velocity dispersion averaged interior to the given radius), or lambdaR{<luminosity>} (computes the $\lambda_R$ statistic of Cappellari et al. 2007)—in the latter three cases {<luminosity>} specifies which band should be used to weight the velocity dispersion, alternatively setting {<luminosity>}=mass (or just leaving off this specifier entirely) will use mass weighting instead.

14.20. Virial Quantities

The following quantities related to the virialized region of each node are output if outputVirialData is set to true:

nodeVirialRadius The virial radius (following whatever definition of virial overdensity was selected in GALACTICUS) in units of Mpc;

nodeVirialVelocity The circular velocity at the virial radius (in km/s).
Part IV.

Development
In this Part we focus on how to modify GALACTICUS to meet your own needs. GALACTICUS is designed in a modular way to make it as simple as possible to introduce new implementations of physical processes or new galactic components without breaking the rest of the code. Nevertheless, some understanding of the structure of the code is necessary. In particular, GALACTICUS will happily compile and run calculations that make no physical sense whatsoever—it’s up to you to ensure that the changes you make are physically reasonable and consistent with the behavior of the rest of the code.
15. Developing **GALACTICUS**

The following is a quickstart guide to making changes to the GALACTICUS source code and contributing them back to the project. Note that the preferred method to do this is through **BitBucket**.

15.1. Getting Started

It’s easy to begin working with and changing the GALACTICUS source code. Assuming you have Mercurial ("hg") installed, just do:

```bash
hg clone https://abensonca@bitbucket.org/abensonca/galacticus galacticus
```

and you have a cloned copy of the GALACTICUS repository in the `galacticus` directory.

15.1.1. Using **BitBucket**

If you plan to contribute changes back to the GALACTICUS project (please do!), you should consider using **BitBucket**. After you’ve created an account for yourself at **BitBucket**, you can “fork” the GALACTICUS repository to have your own working copy. This can be done as follows:

- visit the GALACTICUS repository on **BitBucket** at [https://bitbucket.org/abensonca/galacticus/overview](https://bitbucket.org/abensonca/galacticus/overview);
- click on the “Fork” button—you’ll be presented with a form;
- fill out the form (setting a name for your fork, a short description, etc.), then click the “Fork repository” button;
- after the fork completes, you’ll be taken to the overview page for your forked repository.

You’ll now want to clone this forked repository to your local system. Click on the “Clone” button and copy the `hg clone` command presented there (you want the **SSH** version so that you can push changes back to this repository). Run this command on your local system to get a cloned copy of your new repository. You can now work with this repository, make any changes, and commit them. We’ll discuss how to send these changes back to **BitBucket**, and back to the GALACTICUS project below.

15.2. Making Simple Changes

If you want to make some relatively minor changes to the GALACTICUS code, such as fixing a typo, adding a new filter, etc. you can just make changes directly on the `default` branch (i.e. at the point of active development). To do this, make sure you’re at `default`:

```bash
hg pull
hg update default
```

Then make your changes, add new files, etc. Once you’re done, first check if there have been any changes to `default` since you pulled:

```bash
hg status
```
15. Developing **Galacticus**

hg incoming

If any new changesets are shown, use `hg pull -u` to merge these in to your working copy. Then commit your changes:

`hg commit`

Your changes are now committed to your cloned repository.

15.2.1. Contributing Your Changes Back To **Galacticus**

Once you’ve committed your changes, you can contribute them back to the **Galacticus** project.

**Via E-mail**

If you just cloned the **Galacticus** repository directly you can send a patch containing your changes by e-mail to `abensonobs.carnegiescience.edu`. First, create the patch file using:

```sh
hg export -r begin:end > changes.diff
```

where `begin` and `end` are the first and last revisions that you want to include (you can specify more complicated sets of revisions of course). Then simply attached the `changes.diff` file to an e-mail. It will be merged into the **Galacticus** project using

```sh
hg import changes.diff
```

**Using BitBucket**

If you forked the **Galacticus** repository on BitBucket, you can now push your changes back to BitBucket using

```sh
hg push
```

If you want to contribute these changes back to the **Galacticus** project the best way to do so is to create a “pull request”. Simply visit your forked repository on BitBucket and click the “Pull request” button. The form you’re presented with allows you to choose which branch in your repository you want to send changes from, and which branch in the **Galacticus** project you want them contributed to. Add a title and description of your changes (and, optionally, check the “Close branch” box if you’re done with this branch of development) then click “Create pull request”. Assuming your code looks good and works, it can then be pulled into the **Galacticus** project.

15.3. Making Bigger Changes

For bigger changes, particularly those where you’re adding a new feature, we recommend using Mercurial’s “feature branches”. These provide a permanent record of for which feature each changeset was added. Using feature branches is straightforward. Begin with `default` and create a new branch:

```sh
hg update default
hg branch myNewFeature
```

where `myNewFeature` is a name for your feature branch. Then begin working, make changes, add new files etc. You can make commits when necessary (and it’s good to make several small commits rather than one big one). You should merge `default` into your feature branch as often as possible to avoid them getting out of sync (which makes for difficulty later when you want to merge your feature branch back into `default`):
hg update myFeatureBranch
hg merge default
hg commit -m "merged default into myFeatureBranch"

Once the feature branch is stable, you can merge it back into default:

hg update default
hg merge myFeatureBranch
hg commit -m "merged myFeatureBranch"

Once you’re done developing this feature, you should close the feature branch:

hg commit --close-branch -m "finished my feature"

Note that you can always go back and work on a feature branch later, after you have closed it. Just do:

hg up myFeatureBranch
hg merge default
hg commit -m "merged default into myFeatureBranch"

then continue to work with your feature branch as normal (don’t forget to close it again when you’re finished working with it).

15.4. Releases

Each release of Galacticus exists as a separate branch within the main Galacticus repository. To work with a particular release use

hg update v0.9.2

replacing the version number with whichever version you want. To get back to the development tip use

hg update default

15.4.1. Bug Fixes In Releases

To make a bugfix in a release, simply hg update to that release, fix the bug, and commit your changes. In many cases you’ll want to fix the same bug in later releases and also in default. To do that, just hg update to each branch in turn, use hg merge fixedBranch (where “fixedBranch” is the name of the branch in which you fixed the branch, and then commit the merge. Once the bug has been fixed you can contribute the fix back to the Galacticus project using the methods described above.
16. Coding GALACTICUS

16.1. Numerical Tools

GALACTICUS provides a variety of tools to solve basic numerical problems. These can be found in files source/numerical/*. GALACTICUS makes use of the GNU Scientific Library for many of these tools, but typically provides a higher-level wrapper around those functions, providing a cleaner interface and, in some cases, additional functionality.

16.1.1. Finding Roots of Equations

Tools for solving equations of the form \( f(x) = 0 \) are provided by the rootFinder object (available via the Root_Finder module). Typical use of this object is as follows:

```fortran
! Import the module.
use Root_Finder

! Create a rootFinder object -- make it OpenMP threadprivate so it can be used ! simultaneously by all threads.
type(rootFinder), save :: finder
$omp threadprivate(finder)

! Check if our root finder has been initialized.
if (.not. finder%isInitialized()) then
  ! Specify the function that evaluates f(x).
call finder%rootFunction (myRootFunction)
  ! Specify the type of root-finding algorithm -- this is optional (Brent's ! method will be used by default).
call finder%type (FGSL_Root_fSolver_Brent)
  ! Specify the tolerances to use in finding the root. Both arguments are ! optional -- values of 1.0d-10 will be used for both absolute and relative ! tolerance by default.
call finder%tolerance (toleranceAbsolute, toleranceRelative)
  ! Specify how the initially provided range can be expanded to bracket the ! root. This is optional -- if not provided no range expansion will be attempted.
call finder%rangeExpand &
    & ( &
      & rangeExpandDownward = 0.5d0, &
      & rangeExpandUpward = 2.0d0, &
      & rangeExpandType = rangeExpandMultiplicative, &
      & rangeDownwardLimit = 1.0d-3, &
      & rangeUpwardLimit = 1.0d+3, &
      & rangeExpandDownwardSignExpect = rangeExpandSignExpectPositive, &
      & rangeExpandUpwardSignExpect = rangeExpandSignExpectNegative
    )
end if
```

329
end if
x=finder%find (rootGuess = 1.0d0)
.
.
double precision function myRootFunction(x)
    implicit none
    double precision, intent(in) :: x
    ...
    return
end function myRootFunction

The above example begins by importing the Root_Finder module and then creating a rootFinder object called finder. This is made OpenMP threadprivate so that it may be used simultaneously by all threads. The first step is to initialize finder—the isInitialized method tells us if this has already happened. The most important step is to specify the function that will evaluate f(x). This is done via the rootFunction method—once done, the rootFinder object is marked as initialized (and the isInitialized method will return true). All other initialization steps are optional. In this example, we use the type method to specify that the Brent algorithm should be used for root finding. Any valid GSL-supported root finding algorithm can be used. We then use the tolerance method to specify both the absolute and relative tolerances in the x variable that must be attained to declare the root to be found. Both arguments are optional—default values of 10^{-10} will be used if either tolerance is not specified.

The final step of initialization is to call the rangeExpand method. This specifies how the initial guessed value or range for x should be expanded to bracket the root. If you plan to always specify an initial range, and know that it will always bracket the root, you do not need to specify how the range should be expanded. In this case we’ve specified that range expansion is multiplicative—that is, the lower and upper values of x defining the range will be multiplied by fixed factors until the root is bracketed—via the rangeExpandType=rangeExpandMultiplicative option. Alternatively, additive expansion is possible using rangeExpandType=rangeExpandAdditive. The factors by which to multiply the lower and upper bounds of the range (or the factor to add in the case of additive expansion) are specified by the rangeExpandDownward and rangeExpandUpward options. It is possible to specify absolute lower/upper limits to the range via the rangeDownwardLimit and rangeUpwardLimit options. The range will not be expanded beyond these limits—if the root cannot be bracketed without exceeding these limits an error condition will occur. Finally, it is possible to indicate the expected sign of f(x) at the lower and/or upper limits via the rangeExpandDownwardSignExpect and rangeExpandUpwardSignExpect options. Valid settings are rangeExpandSignExpectNegative, rangeExpandSignExpectPositive, and rangeExpandSignExpectNone (the default—implying that there is no expectation for the sign). If the sign of f(x) is specified, then range expansion will stop once the expected sign is found. This can often improve efficiency, by allowing the range expander to expand the range in only one direction, resulting in a narrower range in which to search for the root.

Finally, we use the find method to return the value of the root. The first argument to find is the name of the function that evaluates f(x). Additionally, we must supply either rootGuess (a scalar value guess to use as the initial value for both the lower and upper values of the range—note that range expansion must be allowed in this case), or rootRange (a two-element array to use as the initial lower and upper values of the range bracketing the root).

The function evaluating f(x) must have a form compatible with that shown for myRootFunction in the above example.
16.2. Computation Dependencies and Data Files

In many situations, some module in Galacticus might want to perform a calculation and then store the results to a file so that they can be reused later. A good example is the CAMB transfer function method (see §13.4.5), which computes a transfer function using CAMB and stores this function in a file so that it can be re-read next time, avoiding the need to recompute the transfer function. A problem arises in such cases as the calculation may depend on the values of parameters (in our example, the transfer function will depend on cosmological parameters for example). We would like to record which parameter values this calculation refers to, perhaps encoding these into the file name, so that we can reuse these data in a future run only if the parameter values are unchanged. Given the modular nature of Galacticus it is impossible to know in advance which parameters will be relevant (e.g. does the cosmological parameter implementation have a parameter that describes a time varying equation of state for dark energy?).

To address this problem, Galacticus provides a mechanism to generate a unique label for a given module. This label encodes the names of modules on which the module depends, and the names and values of any parameters used by those modules. Functions to construct these labels can be generated automatically, and intelligently determine which specific implementation of any method is active in a given run.

To create a unique labelling function, use the `uniqueLabel` directive. For example, in the CAMB transfer function method the following code is used:

```xml
<uniqueLabel>
  <function>Transfer_Function_CAMB_Label</function>
  <ignore>transferFunctionFile</ignore>
  <ignoreRegex>^imf.*</ignoreRegex>
  <hashFile>myExtraFile.txt</hashFile>
</uniqueLabel>
```

This XML block specifies that the labelling function should be called `Transfer_Function_CAMB_Label`. This function will be generated automatically during Galacticus build and will be available from the `Input_Parameters` module. The `ignore` element specifies that the function should not encode the value of the `transferFunctionFile` parameter in the label (in this case, the CAMB method sets the file name directly). Similarly, the `ignoreRegex` element specifies that the function should not encode the value of any parameter which matches the regular expression provided (in this case, any parameter that begins with “imf”). Any number of `ignore` and `ignoreRegex` elements may be present. Finally, a `hashFile` element specifies the name of a file on which this calculation depends, but which will not be detected automatically (see below). An MD5 digest of the file will be included in the label if the `includeSourceDigest` argument is set to true.

The function returns a `type(varying_string)` object with the encoded label. For the CAMB method this might be:

```fortran
write (0,*) char(Transfer_Function_CAMB_Label())
::cosmology.parameters.simple#Omega_b[0.0455]#Omega_Matter[0.2725]#Omega_DE[0.7275]#T_CMB[2.72548]#H_0[70.2]
```

Note that the relevant cosmological parameters have been encoded into the label. If called with `includeVersion=.true.`, then the Galacticus version string is appended to the label:

```fortran
write (0,*) char(Transfer_Function_CAMB_Label(includeVersion=.true.))
::cosmology.parameters.simple#Omega_b[0.0455]#Omega_Matter[0.2725]#Omega_DE[0.7275]#T_CMB[2.72548]#H_0[70.2]_v0.9.3.r2022
```
This can be useful to ensure that a file was generated by the exact same version of GALACTICUS. It is also possible, by setting the optional includeBuild argument to .true., to append full build information (library versions, environment variables) to the label.

By setting the optional includeSourceDigest argument to .true., an MD5 digest of each source file on which the function depends will be incorporated into the label. In addition to source files, any data file referenced by those source files will be included in the digest calculation. Data files are taken to be any file referenced in the data/ directory and having suffix .xml or .hdf5.

Since the returned label can be long and cumbersome, if called with asHash=.true. the labelling function will return an MD5 hash of the label

\[\text{write (0,*) char(Transfer\_Function\_CAMB\_Label(includeVersion=.true.,asHash=.true.))}\]

\[\text{tudE3GawpjAXBzprr5azU0}\]

This hash can be used as part of the name of the file to which data is written.

Finally, each function can return a list of parameter names and values (as an inputParameterList object) which are included in the unique label. These parameters can then be easily written to a generated file using the outputToXML method of the inputParameterList object.

### 16.3. Optimization

In designing GALACTICUS, we opted for simplicity and clarity over speed. However, there are numerous parts of the code where optimization has been performed without a significant loss of clarity. In this section we discuss some of the techniques used.

#### 16.3.1. Unique IDs and Stored Properties

Frequently, a given property of a node may be required in many different aspects of the calculation. For example, the dark matter halo virial radius is used extensively in several distinct calculations within GALACTICUS. Frequently such calculations are performed for the same node, with the same properties several times. Obviously this is inefficient. It can be advantageous in such cases to store the result of a calculation and, if the function is called again with the same unchanged node to simply return the stored value. GALACTICUS facilitates this by two features.

The first feature is the “unique ID”—an integer number assigned to each node in GALACTICUS and which uniquely identifies a node (i.e. no two nodes processed in a GALACTICUS run will have the same unique ID). This number, which can be retrieved using the uniqueID property of a tree node, can be recorded each time a function is called. If called again for a node with the same unique ID as the previous call, the function can simply return the same answer as on the previous call.

The second feature accounts for the fact that the properties of a node will change, so even if a function is called on a node with the same unique ID it may occasionally need to recompute its result. GALACTICUS provides a calculation reset task (see §17.4.3). All such tasks are performed just prior to the computation of derivatives for a node being evolved. A function can register a calculation reset task and use it to flag that it must update its calculations even if called again with the same node.

### 16.4. Mixed Language Coding

It is possible to incorporate C or C++ code into GALACTICUS. The implementation of C++-integration into GALACTICUS is currently only partially complete— if there is a specific function or feature that you are interested in, please contact the authors. For example, GALACTICUS’s ODE solver will fix the properties of a node and then request that derivatives of all properties be computed. Some functions will then be called multiple times for the same node with unchanged properties.
would like to be C++-interoperable contact us. For an example of the various C++ interoperability see the `star_formation.timescales.disks.Baugh2005.cpp` file.

### 16.4.1. Component Property Methods

C++-wrappers are currently automatically built for the “get” method for all real scalar properties defined for GALACTICUS components. To use these wrappers include the following lines:

```cpp
//: ./work/build/objects.nodes.bindings.C.o
#include <objects.nodes.bindings.C.h>
```

The first line adds an explicit dependency on the bindings file, while the second includes it. A component of a given class can be retrieved from a supplied node pointer using:

```cpp
nodeComponent<Class> this<Class>Component (thisNode);
```

Then, real scalar get methods are available using:

```cpp
value=this<Class>Component.<property>();
```

### 16.4.2. Using Functions in C++

Where functions have been made available in C++ the approach is to make an identical interface as in Fortran, as far as is possible. So, for example, getting the expansion factor for some cosmological time would work as follows in the two languages:

**Fortran**

```fortran
use Cosmology_Functions
expansionFactor=Expansion_Factor(cosmologicalTime)
```

**C++**

```cpp
//: ./work/build/cosmology.functions.o
#include <cosmology.functions.h>
expansionFactor=Expansion_Factor(cosmologicalTime)
```

Currently, the only function with C++-wrappers are `Expansion_Factor` and the double precision and integer scalar versions of `Get_Input_Parameter`. Further wrappers will be added as needed or requested.

### 16.4.3. Adding New Implementations in C++

Adding a new implementation of a method is done in a very similar way as in Fortran. For example, the Baugh2005 disk star formation timescale method initializes as follows:

```cpp
// <starFormationTimescaleDisksMethod>
// <unitName>Star_Formation_Timescale_Disk_Baugh2005.Initialize</unitName>
// </starFormationTimescaleDisksMethod>
typedef double (func)(void *thisNode);
void Star_Formation_Timescale_Disk_Baugh2005_Initialize(char *starFormationTimescaleDisksMethod,func **Star_Formation_Timescale_Disk_Get)
{
    char ourName[] = "Baugh2005";
    if (strcmp(starFormationTimescaleDisksMethod,ourName) == 0) {
        *Star_Formation_Timescale_Disk_Get=&Star_Formation_Timescale_Disk_Baugh2005;
    }
}
```
The usual embedded XML directives specifies that this function should be called to initialize the method. The function is passed both the name of the method that has been selected and a function pointer. If the method is matched, the function simply sets the function pointer to point to the specific function implementing the timescale calculation.

16.5. Objects

16.5.1. Enumerations

Enumerations are used to communicate options to many functions in GALACTICUS. All available enumerations, along with their members, are described below.

tfamilly adjustElements

Description: Used to specify how elements should be adjusted when the metallicity of an abundances object is changed.
Provided by: tfamily module tfamily Abundances_Structure
Members: tfamily adjustElementsNone
         tfamily adjustElementsReset
         tfamily adjustElementsUpdate

tfamilly componentType

Description: Used to specify the component(s) to be queried in galactic structure functions.
Provided by: tfamily module tfamily Galactic_Structure_Options
Members: tfamily componentTypeUnknown
         tfamily componentTypeAll
         tfamily componentTypeDisk
         tfamily componentTypeSpheroid
         tfamily componentTypeHotHalo
         tfamily componentTypeDarkHalo
         tfamily componentTypeBlackHole

tfamilly coordinateSystem

Description: Used to specify the coordinate system of the input coordinates in galactic structure functions.
Provided by: tfamily module tfamily Galactic_Structure_Options
Members: tfamily coordinateSystemSpherical
         tfamily coordinateSystemCylindrical
         tfamily coordinateSystemCartesian

tfamilly dataType

Description: Used to specify the type of data being stored in a mergerTreeData structure metadata entry.
Provided by: tfamily module tfamily Merger_Tree_Data_Structure
Members: tfamily dataTypeInteger
         tfamily dataTypeDouble
         tfamily dataTypeText
tfamily extrapolationType
Description: Used to specify the type of extrapolation to use when interpolating in tables.
Provided by: tfamily module tfamily Tables
Members: tfamily extrapolationTypeExtrapolate
tfamily extrapolationTypeFix
tfamily extrapolationTypeAbort

tfamily massDistributionSymmetry
Description: Used to specify the symmetry of massDistribution objects.
Provided by: tfamily module tfamily Mass_Distributions
Members: tfamily massDistributionSymmetryNone
tfamily massDistributionSymmetryCylindrical
tfamily massDistributionSymmetrySpherical

tfamily massType
Description: Used to specify the mass type(s) to be queried in galactic structure functions.
Provided by: tfamily module tfamily Galactic_Structure_Options
Members: tfamily massTypeUnknown
tfamily massTypeAll
tfamily massTypeDark
tfamily massTypeBaryonic
tfamily massTypeGalactic
tfamily massTypeGaseous
tfamily massTypeStellar
tfamily massTypeBlackHole

tfamily metaDataType
Description: Used to specify the type of metadata being stored in a mergerTreeData structure.
Provided by: tfamily module tfamily Merger_Treestructure
Members: tfamily metaDataGeneric
tfamily metaDataCosmology
tfamily metaDataSimulation
tfamily metaDataGroupFinder
tfamily metaDataTreeBuilder
tfamily metaDataProvenance

tfamily metallicityScale
Description: Used to specify the metallicity scale when working with abundances objects.
Provided by: tfamily module tfamily Abundances_Structure
Members: tfamily linearByMass
tfamily linearByNumber
tfamily logarithmicByMassSolar
tfamily logarithmicByNumberSolar
tfamily linearByMassSolar
tfamily linearByNumberSolar
tfamily propertyType

Description: Used to specify properties in a mergerTreeData structure.
Provided by: tfamily module tfamily Merger_Tree_Data_Structure
Members: tfamily propertyTypeTreeNodeIndex
tfamily propertyTypeDescendantIndex
tfamily propertyTypeHostIndex
tfamily propertyTypeRedshift
tfamily propertyTypeScaleFactor
tfamily propertyTypeNodeMass
tfamily propertyTypeParticleCount
tfamily propertyTypePositionX
tfamily propertyTypePositionY
tfamily propertyTypePositionZ
tfamily propertyTypeVelocityX
tfamily propertyTypeVelocityY
tfamily propertyTypeVelocityZ
tfamily propertyTypeSpinX
tfamily propertyTypeSpinY
tfamily propertyTypeSpinZ
tfamily propertyTypeSpin
tfamily propertyTypeAngularMomentumX
tfamily propertyTypeAngularMomentumY
tfamily propertyTypeAngularMomentumZ
tfamily propertyTypeAngularMomentum
tfamily propertyTypeSpecificAngularMomentumX
tfamily propertyTypeSpecificAngularMomentumY
tfamily propertyTypeSpecificAngularMomentumZ
tfamily propertyTypeSpecificAngularMomentum
tfamily propertyTypeHalfMassRadius
tfamily propertyTypeScaleRadius
tfamily propertyTypeParticleIndex
tfamily propertyTypeMostBoundParticleIndex
tfamily propertyTypeSnapshot
tfamily propertyTypeTreeWeight
tfamily propertyTypeVelocityMaximum
tfamily propertyTypeVelocityDispersion

tfamily rangeExpand

Description: Used to specify the way in which the bracketing range should be expanded when searching for roots using a rootFinder object.
Provided by: tfamily module tfamily Root_Finder
Members: tfamily rangeExpandNull
tfamily rangeExpandAdditive
tfamily rangeExpandMultiplicative
16.5. Objects

tfamily rangeExpandSignExpect

Description: Used to specify the expected sign of the root function when searching for roots using a rootFinder object.

Provided by: tfamily module tfamily Root_Finder

Members: tfamily rangeExpandSignExpectNegative
          tfamily rangeExpandSignExpectNone
          tfamily rangeExpandSignExpectPositive

tfamily units

Description: Used to specify the type of units being stored in a mergerTreeData structure.

Provided by: tfamily module tfamily Merger_Tree_Data_Structure

Members: tfamily unitsStandard
          tfamily unitsTime
          tfamily unitsLittleH
          tfamily unitsMass
          tfamily unitsLength
          tfamily unitsTime
          tfamily unitsVelocity

tfamily weightBy

Description: Used to specify by which quantity to weight the results in galactic structure functions.

Provided by: tfamily module tfamily Galactic_Structure_Options

Members: tfamily weightByMass
          tfamily weightByLuminosity

16.5.2. Object Methods

The type of each method, and the type and names of its arguments are specified for each method of each object. Types are shown in red, enclosed by angle brackets, with a “*” indicating a pointer. A <void> type indicates a subroutine. Blue arrows after each argument show the argument intent: ← implies intent(in), → implies intent(out), and ↔ implies intent(inout).

tfamily abundances

add(<type(abundances)> abundances2 →) Add two abundances.

builder(<*type(node)> abundancesDefinition→) Build an abundances object from a provided XML description.

deserialize(<double(:)> historyArray→) Deserialize an abundances object from an array.

destroy() Destroy an abundances object.

divide(<double> divisor→) Divide an abundance by a scalar.

dump() Dump an abundances object.

dumpRaw(<integer> fileHandle→) Dump an abundances object to binary.
tfamily double heliumMassFraction() Returns the helium fraction by mass.

tfamily double heliumNumberFraction() Returns the helium fraction by number.

tfamily double hydrogenMassFraction() Returns the hydrogen fraction by mass.

tfamily double hydrogenNumberFraction() Returns the hydrogen fraction by number.

tfamily void increment(<type(abundances)> addAbundances→) Increment an abundances object.

tfamily logical isZero() Return true if an abundances object is zero.

tfamily void massToMassFraction(<double> mass→) Converts abundance masses to mass fractions by dividing by the given mass while ensuring that fractions are in the range 0–1.

tfamily double metallicity(<metallicityScale> [metallicityType]→) Returns the metallicity.

tfamily void metallicitySet(<double> metallicity→, <metallicityScale> [metallicityType]→, <adjustElements> [adjustElements]→, <integer> [abundanceIndex]→) Sets the metallicity to metallicity.

tfamily textcolor{red}{\textless type(abundances)\textgreater} multiply(<double> multiplier→) Multiply an abundance by a scalar.

tfamily void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(;;)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount→, <double(;;)> doubleBuffer↔, <double> time→, <integer> instance→) Store an abundances object in the output buffers.

tfamily void outputCount(<integer>integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Specify the count of an abundances object for output.

tfamily void outputNames(<integer>integerProperty↔, <char[*](:)>integerPropertyNames↔, <char[*](:)> integerPropertyComments↔, <double(:)>integerPropertyUnitsSI↔, <integer> doubleProperty↔, <char[*](:)> doublePropertyNames↔, <char[*](:)> doublePropertyComments↔, <double(:)> doublePropertyUnitsSI↔, <double> time→, <integer> instance→) Specify the names of abundance object properties for output.

tfamily void readRaw(<integer> fileHandle→) Read an abundances object from binary.

tfamily void reset() Reset an abundances object.

tfamily void serialize(<double(;;)> historyArray←) Serialize an abundances object to an array.

tfamily int serializeCount() Return a count of the number of properties in a serialized abundances object.

tfamily void setToUnity() Set an abundances object to unity.

tfamily textcolor{red}{\textless type(abundances)\textgreater} subtract(<type(abundances)> abundances2→) Subtract one abundance from another.

tfamily accretionDiskSpectraClass

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision spectrum(<double precision> wavelength→) Returns the spectrum (in units of \(L_{\odot} \text{Hz}^{-1}\)) of the accretion disk at the given wavelength (in units of Å) for node.
tfamily accretionDiskSpectraFile

tfamily logical isFinalizable() Return true if this object can be finalized.

void loadFile(<character(len=*>) fileName) Load a file of AGN spectra.

double precision spectrum(<double precision> wavelength) Returns the spectrum (in units of $L_\odot$ Hz$^{-1}$) of the accretion disk at the given wavelength (in units of Å) for node.

---

tfamily accretionDiskSpectraHopkins2007

tfamily logical isFinalizable() Return true if this object can be finalized.

void loadFile(<character(len=*>) fileName) Load a file of AGN spectra.

double precision spectrum(<double precision> wavelength) Returns the spectrum (in units of $L_\odot$ Hz$^{-1}$) of the accretion disk at the given wavelength (in units of Å) for node.

---

tfamily accretionHaloClass

tfamily double precision accretedMass(<integer> accretionMode) Returns the mass (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily type(chemicalAbundances) accretedMassChemicals(<integer> accretionMode) Returns the mass of chemicals (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily type(abundances) accretedMassMetals(<integer> accretionMode) Returns the mass of metals (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily double precision accretionRate(<integer> accretionMode) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of mass from the IGM onto node in the given accretionMode.

tfamily type(chemicalAbundances) accretionRateChemicals(<integer> accretionMode) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of chemicals from the IGM onto node in the given accretionMode.

tfamily type(abundances) accretionRateMetals(<integer> accretionMode) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of metals from the IGM onto node in the given accretionMode.

tfamily double precision failedAccretedMass(<integer> accretionMode) Returns the mass (in units of $M_\odot$) of that failed to accrete from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily double precision failedAccretionRate(<integer> accretionMode) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of failed accretion of mass from the IGM onto node in the given accretionMode.

logical isFinalizable() Return true if this object can be finalized.
tfamily accretionHaloColdMode

tfamily double precision accretedMass(<integer> accretionMode→) Returns the mass (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily type(chemicalAbundances) accretedMassChemicals(<integer> accretionMode→) Returns the mass of chemicals (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily type(abundances) accretedMassMetals(<integer> accretionMode→) Returns the mass of metals (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily double precision accretionRate(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of mass from the IGM onto node in the given accretionMode.

tfamily type(chemicalAbundances) accretionRateChemicals(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of chemicals from the IGM onto node in the given accretionMode.

tfamily type(abundances) accretionRateMetals(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of metals from the IGM onto node in the given accretionMode.

tfamily double precision accretionRateTotal(<type(treeNode>) *node juvenescence) Returns the total accretion rate from the IGM onto a halo (including dark matter).

tfamily \textcolor{red}{\textless\textgreater} chemicalMasses(<type(treeNode>) *node juvenescence), <double> massAccreted →, <integer> accretionMode→) Returns the total accretion rate from the IGM onto a halo (including dark matter).

tfamily double zero coldModeFraction(<type(treeNode)> *node juvenescence, <integer> accretionMode→) Returns the total accretion rate from the IGM onto a halo (including dark matter).

tfamily double precision failedAccretedMass(<integer> accretionMode→) Returns the mass (in units of $M_\odot$) of that failed to accrete from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily double precision failedAccretionRate(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of failed accretion of mass from the IGM onto node in the given accretionMode.

tfamily double precision failedFraction(<type(treeNode)> *node juvenescence) Returns the fraction of potential accretion onto a halo from the IGM which fails.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision massTotal(<type(treeNode)> *node juvenescence) Returns the total node mass.

tfamily double precision velocityScale(<type(treeNode)> *node juvenescence) Returns the velocity scale to use for node.
tfamily accretionHaloNaozBarkana2007

tfamily double precision accretedMass(<integer> accretionMode→) Returns the mass (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily type(chemicalAbundances) accretedMassChemicals(<integer> accretionMode→) Returns the mass of chemicals (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily type(abundances) accretedMassMetals(<integer> accretionMode→) Returns the mass of metals (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.

 tfamily double precision accretionRate(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of mass from the IGM onto node in the given accretionMode.

tfamily type(chemicalAbundances) accretionRateChemicals(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of chemicals from the IGM onto node in the given accretionMode.

tfamily type(abundances) accretionRateMetals(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of metals from the IGM onto node in the given accretionMode.

 tfamily double precision accretionRateTotal(<type(treeNode)> *node→) Returns the total accretion rate from the IGM onto a halo (including dark matter).

tfamily double precision failedAccretedMass(<integer> accretionMode→) Returns the mass (in units of $M_\odot$) of that failed to accrete from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily double precision failedAccretionRate(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of failed accretion of mass from the IGM onto node in the given accretionMode.

tfamily double precision failedFraction(<type(treeNode)> *node→) Returns the fraction of potential accretion onto a halo from the IGM which fails.

 tfamily logical isFinalizable() Return true if this object can be finalized..

tfamily double precision massTotal(<type(treeNode)> *node→) Returns the total node mass.

tfamily double precision velocityScale(<type(treeNode)> *node→) Returns the velocity scale to use for node.

tfamily accretionHaloNull

 tfamily double precision accretedMass(<integer> accretionMode→) Returns the mass (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily type(chemicalAbundances) accretedMassChemicals(<integer> accretionMode→) Returns the mass of chemicals (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily type(abundances) accretedMassMetals(<integer> accretionMode→) Returns the mass of metals (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.
16. Coding GALACTICUS

tfamily double precision accretionRate(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of mass from the IGM onto node in the given accretionMode.

tfamily type(chemicalAbundances) accretionRateChemicals(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of chemicals from the IGM onto node in the given accretionMode.

tfamily type(abundances) accretionRateMetals(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of metals from the IGM onto node in the given accretionMode.

tfamily double precision failedAccretedMass(<integer> accretionMode→) Returns the mass (in units of $M_\odot$) of that failed to accrete from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily double precision failedAccretionRate(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of failed accretion of mass from the IGM onto node in the given accretionMode.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily accretionHaloSimple

tfamily double precision accretedMass(<integer> accretionMode→) Returns the mass (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily type(chemicalAbundances) accretedMassChemicals(<integer> accretionMode→) Returns the mass of chemicals (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily type(abundances) accretedMassMetals(<integer> accretionMode→) Returns the mass of metals (in units of $M_\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily double precision accretionRate(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of mass from the IGM onto node in the given accretionMode.

tfamily type(chemicalAbundances) accretionRateChemicals(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of chemicals from the IGM onto node in the given accretionMode.

tfamily type(abundances) accretionRateMetals(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of metals from the IGM onto node in the given accretionMode.

tfamily double precision accretionRateTotal(<type(treeNode>) *node→) Returns the total accretion rate from the IGM onto a halo (including dark matter).

tfamily double precision failedAccretedMass(<integer> accretionMode→) Returns the mass (in units of $M_\odot$) of that failed to accrete from the IGM onto node in the given accretionMode. Used to initialize nodes.

tfamily double precision failedAccretionRate(<integer> accretionMode→) Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of failed accretion of mass from the IGM onto node in the given accretionMode.

tfamily double precision failedFraction(<type(treeNode>) *node→) Returns the fraction of potential accretion onto a halo from the IGM which fails.
16.5. Objects

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision massTotal(<type(treeNode)> *node) Returns the total node mass.

tfamily double precision velocityScale(<type(treeNode)> *node) Returns the velocity scale to use for node.

tfamily chemicalAbundances

tfamily \ doublezero abundance(<integer> moleculeIndex) Returns the abundance of a chemical given its index.

tfamily \ void abundanceSet(<integer> moleculeIndex, <double> abundance) Sets the abundance of a chemical given its index.

tfamily \ textcolor{red}{\textless type(chemicalAbundances)\textgreater} add(<type(chemicalAbundances)> abundances2) Add two chemical abundances.

tfamily \ void builder(<*type(node)> chemicalAbundancesDefinition) Build a chemical abundances object from an XML definition.

tfamily \ void deserialize(<double(:)> chemicalAbundancesArray) Deserialize a chemical abundances object from an array.

tfamily \ void destroy() Destroys a chemical abundances object.

tfamily \ textcolor{red}{\textless type(chemicalAbundances)\textgreater} divide(<double> divisor) Divide a chemical abundance by a scalar.

tfamily \ void dump() Dump a chemical abundances object.

tfamily \ void dumpRaw(<integer> fileHandle) Dump a chemical abundances object in binary.

tfamily \ void enforcePositive() Enforces all chemical values to be positive.

tfamily \ void increment(<type(chemicalAbundances)> addAbundances) Increment a chemical abundances object.

tfamily \ logicalzero isZero() Return true if a chemicals object is zero.

tfamily \ void massToNumber(<type(chemicalAbundances)> chemicalsByNumber) Converts from abundances by mass to abundances by number.

tfamily \ textcolor{red}{\textless type(chemicalAbundances)\textgreater} multiply(<double> multiplier) Multiply a chemical abundance by a scalar.

tfamily \ void numberToMass(<type(chemicalAbundances)> chemicalsByMass) Converts from abundances by number to abundances by mass.

tfamily \ void readRaw(<integer> fileHandle) Read a chemical abundances object in binary.

tfamily \ void reset() Resets abundances to zero.

tfamily \ void serialize(<double(:)> chemicalAbundancesArray) Serialize a chemical abundances object to an array.
16. Coding GALACTICUS

```c
\textfamily intzero serializeCount() \text{return a count of the number of properties in a serialized chemical abundances object.}
\textfamily void setToUnity() \text{Set abundances to unity.}
\textfamily \textcolor{red}{\textless\textbf{type(chemicalAbundances)}\textgreater} subtract(<\textbf{type(chemicalAbundances)}> abundances2\rightarrow) \text{return one chemical abundance from another.}

\textfamily chemicalStructure
\textfamily intzero charge() \text{return the charge of a chemical.}
\textfamily void export(<\text{character(len=*)}> coutputFile\rightarrow) \text{Write a chemical structure to a CML file.}
\textfamily doublezero mass() \text{return the mass of a chemical in atomic mass units.}
\textfamily void retrieve(<\text{character(len=*)}> chemicalName\rightarrow) \text{Get a chemical from the database.}

\textfamily coordinate
\textfamily void fromCartesian(<\textbf{double(3)}> x\rightarrow) \text{Set the coordinates from a Cartesian system specified as a 3-element array.}
\textfamily \textcolor{red}{\textless\textbf{double(3)}\textgreater} toCartesian() \text{return the coordinates in a Cartesian system as a 3-element array.}

\textfamily coordinateCartesian
\textfamily void fromCartesian(<\textbf{double(3)}> x\rightarrow) \text{Set the coordinates from a Cartesian system specified as a 3-element array.}
\textfamily \textcolor{red}{\textless\textbf{double(3)}\textgreater} toCartesian() \text{return the coordinates in a Cartesian system as a 3-element array.}
\textfamily doublezero x() \text{return the x-coordinate.}
\textfamily void xSet(<\textbf{double}> x\rightarrow) \text{set the x-coordinate.}
\textfamily doublezero y() \text{return the y-coordinate.}
\textfamily void ySet(<\textbf{double}> y\rightarrow) \text{set the y-coordinate.}
\textfamily doublezero z() \text{return the z-coordinate.}
\textfamily void zSet(<\textbf{double}> z\rightarrow) \text{set the z-coordinate.}

\textfamily coordinateCylindrical
\textfamily void fromCartesian(<\textbf{double(3)}> x\rightarrow) \text{Set the coordinates from a Cartesian system specified as a 3-element array.}
\textfamily doublezero phi() \text{return the φ-coordinate.}
\textfamily void phiSet(<\textbf{double}> phi\rightarrow) \text{set the φ-coordinate.}
\textfamily doublezero r() \text{return the r-coordinate.}
```
16.5. Objects

tfamily \void rSet(<double> r \rightarrow) set the r-coordinate.

tfamily \textcolor{red}{\textless double(3)\textgreater} toCartesian() Return the coordinates in a Cartesian system as a 3-element array.

tfamily \doublezero z() Get the z-coordinate.

tfamily \void zSet(<double> z \rightarrow) set the z-coordinate.

tfamily coordinateSpherical

tfamily \void fromCartesian(<double(3)> x \rightarrow) Set the coordinates from a Cartesian system specified as a 3-element array.

tfamily \doublezero phi() Get the φ-coordinate.

tfamily \void phiSet(<double> phi \rightarrow) set the φ-coordinate.

tfamily \doublezero r() Get the r-coordinate.

tfamily \void rSet(<double> r \rightarrow) set the r-coordinate.

tfamily \doublezero theta() Get the θ-coordinate.

tfamily \void thetaSet(<double> theta \rightarrow) set the θ-coordinate.

tfamily \textcolor{red}{\textless double(3)\textgreater} toCartesian() Return the coordinates in a Cartesian system as a 3-element array.

tfamily cosmologyFunctionsClass

tfamily double precision comovingVolumeElementRedshift(<double precision> time \rightarrow) Returns the differential comoving volume element \( dV/dz = r_c^2(t)cH^{-1}(t) \) (where \( r_c \) is the comoving distance to time \( t \) and \( H(t) \) is the Hubble parameter at that time) for unit solid angle at the specified time.

tfamily double precision comovingVolumeElementTime(<double precision> time \rightarrow) Returns the differential comoving volume element \( dV/dt = r_c^2(t)ca(t) \) (where \( r_c \) is the comoving distance to time \( t \) and \( a(t) \) is the expansion at that time) for unit solid angle at the specified time.

tfamily double precision cosmicTime(<double precision> expansionFactor \rightarrow, <logical> collapsingPhase \rightarrow) Return the cosmological age at the given expansion factor.

tfamily logical cosmicTimeIsValid(<double precision> time \rightarrow) Returns true if the given cosmic time is valid one for this cosmology.

tfamily void densityScalingEarlyTime(<double precision> dominateFactor \rightarrow, <double precision> densityPower ←, <double precision> expansionFactorDominant ←, <double precision> OmegaDominant ←) Compute the scaling of density with expansion factor at early times in the universe.

tfamily double precision distanceAngular(<double precision> time \rightarrow) Return the angular diameter distance to the given cosmic time.

tfamily double precision distanceComoving(<double precision> time \rightarrow) Return the comoving distance to the given cosmic time.
tfamily double precision distanceComovingConvert(<integer> output→,<double precision> distance-Modulus→,<double precision> redshift→) Convert between different measures of comoving distance.

tfamily double precision distanceLuminosity(<double precision> time→) Return the luminosity distance to the given cosmic time.

tfamily double precision dominationEpochMatter(<double precision> dominateFactor→) Compute the epoch at which matter dominates over other forms of energy by a given factor.

tfamily double precision epochTime(<double precision> time→,<double precision> expansionFactor→,<logical> collapsingPhase→) Convenience function that returns the time corresponding to an epoch specified by time or expansion factor.

tfamily double precision equalityEpochMatterCurvature(<integer> requestType→) Return the epoch of matter-curvature magnitude equality (either expansion factor or cosmic time).

tfamily double precision equalityEpochMatterDarkEnergy(<integer> requestType→) Return the epoch of matter-dark energy magnitude equality (either expansion factor or cosmic time).

tfamily double precision equationOfStateDarkEnergy(<double precision> time→,<double precision> expansionFactor→) Returns the cosmological expansion rate, $\dot{a}/a$ at expansion factor expansionFactor.

tfamily double precision exponentDarkEnergy(<double precision> time→,<double precision> expansionFactor→) HASH(0x198caa8)

tfamily double precision hubbleParameterEpochal(<double precision> time→,<double precision> expansionFactor→,<logical> collapsingPhase→) Returns the Hubble parameter at the requested cosmological time, time, or expansion factor, expansionFactor.

tfamily double precision hubbleParameterRateOfChange(<double precision> time→,<double precision> expansionFactor→,<logical> collapsingPhase→) Returns the rate of change of the Hubble parameter at the requested cosmological time, time, or expansion factor, expansionFactor.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision omegaDarkEnergyEpochal(<double precision> time→,<double precision> expansionFactor→,<logical> collapsingPhase→) Return the dark energy density parameter at expansion factor expansionFactor.

tfamily double precision omegaMatterEpochal(<double precision> time→,<double precision> expansionFactor→,<logical> collapsingPhase→) Return the matter density parameter at expansion factor expansionFactor.
16.5. Objects

**Object Methods**

- `tfamily double precision omegaMatterRateOfChange(<double precision> time→, <double precision> expansionFactor→, <logical> collapsingPhase→)` Return the rate of change of the matter density parameter at expansion factor `expansionFactor`.

- `tfamily double precision redshiftFromExpansionFactor(<double precision> expansionFactor→)` Returns redshift for a given expansion factor.

- `tfamily void stateRestore(<integer> stateFile→)` Restore the state of the object to file.

- `tfamily void stateStore(<integer> stateFile→)` Store the state of the object to file.

- `tfamily double precision temperatureCMBEpochal(<double precision> time→, <double precision> expansionFactor→, <logical> collapsingPhase→)` Return the temperature of the cosmic microwave background at `expansionFactor`.

- `tfamily double precision timeAtDistanceComoving(<double precision> comovingDistance→)` Return the cosmic time corresponding to the given `comovingDistance`.

- `tfamily double precision comovingVolumeElementRedshift(<double precision> time→)` Returns the differential comoving volume element $\frac{dV}{dz} = r_c^2(t)cH(t)^{-1}$ (where $r_c$ is the comoving distance to time $t$ and $H(t)$ is the Hubble parameter at that time) for unit solid angle at the specified `time`.

- `tfamily double precision comovingVolumeElementTime(<double precision> time→)` Returns the differential comoving volume element $\frac{dV}{dt} = r_c^2(t)ca(t)$ (where $r_c$ is the comoving distance to time $t$ and $a(t)$ is the expansion at that time) for unit solid angle at the specified `time`.

- `tfamily double precision cosmicTime(<double precision> expansionFactor→, <logical> collapsingPhase→)` Return the cosmological age at the given expansion factor.

- `tfamily logical cosmicTimeIsValid(<double precision> time→)` Returns true if the given cosmic time is valid one for this cosmology.

- `tfamily void densityScalingEarlyTime(<double precision> dominateFactor→, <double precision> densityPower←, <double precision> expansionFactorDominant←, <double precision> OmegaDominant←)` Compute the scaling of density with expansion factor at early times in the universe.

- `tfamily double precision distanceAngular(<double precision> time→)` Return the angular diameter distance to the given cosmic `time`.

- `tfamily double precision distanceComoving(<double precision> time→)` Return the comoving distance to the given cosmic `time`.

- `tfamily double precision distanceComovingConvert(<integer> output→, <double precision> distanceModulus→, <double precision> redshift→)` Convert between different measures of comoving distance.

- `tfamily double precision distanceLuminosity(<double precision> time→)` Return the luminosity distance to the given cosmic `time`.

- `tfamily void distanceTabulate(<double> time→)` Tabulate comoving distance as a function of cosmic `time`.

- `tfamily double precision dominationEpochMatter(<double precision> dominateFactor→)` Compute the epoch at which matter dominates over other forms of energy by a given factor.
16. Coding GALACTICUS

tfamily double precision epochTime(\texttt{<double precision> time\rightarrow}, \texttt{<double precision> expansionFactor\rightarrow}, \texttt{<logical> collapsingPhase\rightarrow}) Convenience function that returns the time corresponding to an epoch specified by time or expansion factor.

tfamily double precision equalityEpochMatterCurvature(\texttt{<integer> requestType\rightarrow}) Return the epoch of matter-curvature magnitude equality (either expansion factor or cosmic time).

tfamily double precision equalityEpochMatterDarkEnergy(\texttt{<integer> requestType\rightarrow}) Return the epoch of matter-dark energy magnitude equality (either expansion factor or cosmic time).

tfamily double precision equationOfStateDarkEnergy(\texttt{<double precision> time\rightarrow}, \texttt{<double precision> expansionFactor\rightarrow}) Returns the dark energy density parameter at expansion factor.

tfamily double precision expansionFactor(\texttt{<double precision> time\rightarrow}) Returns the expansion factor at cosmological time \texttt{time}.

tfamily double precision expansionFactorFromRedshift(\texttt{<double precision> redshift\rightarrow}) Returns expansion factor given a redshift.

tfamily logical expansionFactorIsValid(\texttt{<double precision> expansionFactor\rightarrow}) Returns true if the given expansion factor is valid one for this cosmology.

tfamily void expansionFactorTabulate(\texttt{<double> time\rightarrow}) Tabulate expansion factor as a function of cosmic time.

tfamily double precision expansionRate(\texttt{<double precision> expansionFactor\rightarrow}) Returns the cosmological expansion rate, $\dot{a}/a$ at expansion factor \texttt{expansionFactor}.

tfamily double precision hubbleParameterEpochal(\texttt{<double precision> time\rightarrow}, \texttt{<double precision> expansionFactor\rightarrow}, \texttt{<logical> collapsingPhase\rightarrow}) Returns the Hubble parameter at the requested cosmological time, \texttt{time}, or expansion factor, \texttt{expansionFactor}.

tfamily double precision hubbleParameterRateOfChange(\texttt{<double precision> time\rightarrow}, \texttt{<double precision> expansionFactor\rightarrow}, \texttt{<logical> collapsingPhase\rightarrow}) Returns the rate of change of the Hubble parameter at the requested cosmological time, \texttt{time}, or expansion factor, \texttt{expansionFactor}.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision omegaDarkEnergyEpochal(\texttt{<double precision> time\rightarrow}, \texttt{<double precision> expansionFactor\rightarrow}, \texttt{<logical> collapsingPhase\rightarrow}) Return the dark energy density parameter at expansion factor \texttt{expansionFactor}.

tfamily double precision omegaMatterEpochal(\texttt{<double precision> time\rightarrow}, \texttt{<double precision> expansionFactor\rightarrow}, \texttt{<logical> collapsingPhase\rightarrow}) Return the matter density parameter at expansion factor \texttt{expansionFactor}.

tfamily double precision omegaMatterRateOfChange(\texttt{<double precision> time\rightarrow}, \texttt{<double precision> expansionFactor\rightarrow}, \texttt{<logical> collapsingPhase\rightarrow}) Return the rate of change of the matter density parameter at expansion factor \texttt{expansionFactor}.
tfamily double precision redshiftFromExpansionFactor(<double precision> expansionFactor→) Returns redshift for a given expansion factor.

tfamily void stateRestore(<integer> stateFile→) Restore the state of the object to file.

tfamily void stateStore(<integer> stateFile→) Store the state of the object to file.

tfamily void targetSelf() Set a module-scope pointer to self.

tfamily double precision temperatureCMBEpochal(<double precision> time→,<double precision> expansionFactor→,<logical> collapsingPhase→) Return the temperature of the cosmic microwave background at expansionFactor.

tfamily double precision timeAtDistanceComoving(<double precision> comovingDistance→) Return the cosmic time corresponding to the given comovingDistance.

tfamily cosmologyFunctionsMatterLambda

tfamily double precision comovingVolumeElementRedshift(<double precision> time→) Returns the differential comoving volume element \(dV/dz = r_c^2(t)H^{-1}(t)\) (where \(r_c\) is the comoving distance to time \(t\) and \(H(t)\) is the Hubble parameter at that time) for unit solid angle at the specified time.

tfamily double precision comovingVolumeElementTime(<double precision> time→) Returns the differential comoving volume element \(dV/dt = r_c^2(t)ca(t)\) (where \(r_c\) is the comoving distance to time \(t\) and \(a(t)\) is the expansion at that time) for unit solid angle at the specified time.

tfamily double precision cosmicTime(<double precision> expansionFactor→,<logical> collapsingPhase→) Return the cosmological age at the given expansion factor.

tfamily logical cosmicTimeIsValid(<double precision> time→) Returns true if the given cosmic time is valid one for this cosmology.

tfamily void densityScalingEarlyTime(<double precision> dominateFactor→,<double precision> densityPower←,<double precision> expansionFactorDominant←,<double precision> OmegaDominant←) Compute the scaling of density with expansion factor at early times in the universe.

tfamily double precision distanceAngular(<double precision> time→) Return the angular diameter distance to the given cosmic time.

tfamily double precision distanceComoving(<double precision> time→) Return the comoving distance to the given cosmic time.

tfamily double precision distanceComovingConvert(<integer> output→,<double precision> distanceModulus→,<double precision> redshift→) Convert between different measures of comoving distance.

tfamily double precision distanceLuminosity(<double precision> time→) Return the luminosity distance to the given cosmic time.

tfamily void distanceTabulate(<double> time→) Tabulate comoving distance as a function of cosmic time.

tfamily double precision dominationEpochMatter(<double precision> dominateFactor→) Compute the epoch at which matter dominates over other forms of energy by a given factor.
16. Coding Galacticus

tfamily double precision epochTime(<double precision> time→, <double precision> expansionFactor→, <logical> collapsingPhase→) Convenience function that returns the time corresponding to an epoch specified by time or expansion factor.

tfamily double precision equalityEpochMatterCurvature(<integer> requestType→) Return the epoch of matter-curvature magnitude equality (either expansion factor or cosmic time).

tfamily double precision equalityEpochMatterDarkEnergy(<integer> requestType→) Return the epoch of matter-dark energy magnitude equality (either expansion factor or cosmic time).

tfamily double precision equationOfStateDarkEnergy(<double precision> time→, <double precision> expansionFactor→) HASH(0x1989a30)

tfamily double precision expansionFactor(<double precision> time→) Returns the expansion factor at cosmological time time.

tfamily double precision expansionFactorFromRedshift(<double precision> redshift→) Returns expansion factor given a redshift.

tfamily logical expansionFactorIsValid(<double precision> expansionFactor→) Returns true if the given expansion factor is valid one for this cosmology.

tfamily void expansionFactorTabulate(<double> time→) Tabulate expansion factor as a function of cosmic time.

tfamily double precision expansionRate(<double precision> expansionFactor→) Returns the cosmological expansion rate, \( \dot{a}/a \) at expansion factor expansionFactor.

tfamily double precision exponentDarkEnergy(<double precision> time→, <double precision> expansionFactor→) HASH(0x198caa8)

tfamily double precision hubbleParameterEpochal(<double precision> time→, <double precision> expansionFactor→, <logical> collapsingPhase→) Returns the Hubble parameter at the requested cosmological time, time, or expansion factor, expansionFactor.

tfamily double precision hubbleParameterRateOfChange(<double precision> time→, <double precision> expansionFactor→, <logical> collapsingPhase→) Returns the rate of change of the Hubble parameter at the requested cosmological time, time, or expansion factor, expansionFactor.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision omegaDarkEnergyEpochal(<double precision> time→, <double precision> expansionFactor→, <logical> collapsingPhase→) Return the dark energy density parameter at expansion factor expansionFactor.

tfamily double precision omegaMatterEpochal(<double precision> time→, <double precision> expansionFactor→, <logical> collapsingPhase→) Return the matter density parameter at expansion factor expansionFactor.

tfamily double precision omegaMatterRateOfChange(<double precision> time→, <double precision> expansionFactor→, <logical> collapsingPhase→) Return the rate of change of the matter density parameter at expansion factor expansionFactor.

tfamily double precision redshiftFromExpansionFactor(<double precision> expansionFactor→) Returns redshift for a given expansion factor.
tfamily void stateRestore(<integer> stateFile→) Restore the state of the object to file.

tfamily void stateStore(<integer> stateFile→) Store the state of the object to file.

tfamily double precision temperatureCMBEpochal(<double precision> time→,<double precision> expansionFactor→,<logical> collapsingPhase→) Return the temperature of the cosmic microwave background at \texttt{expansionFactor}.


tfamily double precision timeAtDistanceComoving(<double precision> comovingDistance→) Return the cosmic time corresponding to the given \texttt{comovingDistance}.


tfamily cosmologyParametersClass
tfamily double precision densityCritical() Return the critical density at the present day in units of \(M_\odot/\text{Mpc}^3\).

tfamily double precision HubbleConstant(<integer> units→) Return the Hubble constant at the present day. The optional \texttt{units} argument specifies if the return value should be in units of \(\text{km}/\text{s}/\text{Mpc}\) (\texttt{unitsStandard}), \(\text{Gyr}^{-1}\) (\texttt{unitsTime}), or 100 \(\text{km}/\text{s}/\text{Mpc}\) (\texttt{unitsLittleH}).

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision OmegaBaryon() Return the cosmological baryon density in units of the critical density at the present day.

tfamily double precision OmegaCurvature() Return the cosmological curvature density in units of the critical density at the present day.

tfamily double precision OmegaDarkEnergy() Return the cosmological dark energy density in units of the critical density at the present day.

tfamily double precision OmegaMatter() Return the cosmological matter density in units of the critical density at the present day.

tfamily double precision OmegaRadiation() Return the cosmological radiation density in units of the critical density at the present day.

tfamily double precision temperatureCMB() Return the temperature of the cosmic microwave background radiation (in units of Kelvin) at the present day.


tfamily cosmologyParametersSimple
tfamily double precision densityCritical() Return the critical density at the present day in units of \(M_\odot/\text{Mpc}^3\).

tfamily double precision HubbleConstant(<integer> units→) Return the Hubble constant at the present day. The optional \texttt{units} argument specifies if the return value should be in units of \(\text{km}/\text{s}/\text{Mpc}\) (\texttt{unitsStandard}), \(\text{Gyr}^{-1}\) (\texttt{unitsTime}), or 100 \(\text{km}/\text{s}/\text{Mpc}\) (\texttt{unitsLittleH}).

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision OmegaBaryon() Return the cosmological baryon density in units of the critical density at the present day.

tfamily double precision OmegaCurvature() Return the cosmological curvature density in units of the critical density at the present day.

351
16. Coding GALACTICUS

tfamily double precision OmegaDarkEnergy() Return the cosmological dark energy density in units of the critical density at the present day.

tfamily double precision OmegaMatter() Return the cosmological matter density in units of the critical density at the present day.

tfamily double precision OmegaRadiation() Return the cosmological radiation density in units of the critical density at the present day.

tfamily double precision temperatureCMB() Return the temperature of the cosmic microwave background radiation (in units of Kelvin) at the present day.

tfamily darkMatterHaloScaleClass
tfamily void calculationReset() Reset the calculation state of the object.

tfamily double precision dynamicalTimescale() The characteristic dynamical timescale of a dark matter halo.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision meanDensity() The mean density of a dark matter halo.

tfamily double precision meanDensityGrowthRate() The growth rate of the mean density of a dark matter halo.

tfamily void stateRestore(<integer> stateFile →) Restore the state of the object to file.

tfamily void stateStore(<integer> stateFile →) Store the state of the object to file.

tfamily double precision virialRadius() The virial radius of a dark matter halo.

tfamily double precision virialRadiusGrowthRate() The growth rate of the virial radius of a dark matter halo.

tfamily double precision virialTemperature() The virial temperature of a dark matter halo.

tfamily double precision virialVelocity() The virial velocity of a dark matter halo.

tfamily double precision virialVelocityGrowthRate() The growth rate of the virial velocity of a dark matter halo.

tfamily darkMatterHaloScaleVirialDensityContrastDefinition
tfamily void calculationReset() Reset the calculation state of the object.

tfamily double precision dynamicalTimescale() The characteristic dynamical timescale of a dark matter halo.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision meanDensity() The mean density of a dark matter halo.

tfamily double precision meanDensityGrowthRate() The growth rate of the mean density of a dark matter halo.

tfamily void stateRestore(<integer> stateFile →) Restore the state of the object to file.
16.5. Objects

tfamily void stateStore(<integer> stateFile→) Store the state of the object to file.

tfamily double precision virialRadius() The virial radius of a dark matter halo.

tfamily double precision virialRadiusGrowthRate() The growth rate of the virial radius of a dark matter halo.

tfamily double precision virialTemperature() The virial temperature of a dark matter halo.

tfamily double precision virialVelocity() The virial velocity of a dark matter halo.

tfamily double precision virialVelocityGrowthRate() The growth rate of the virial velocity of a dark matter halo.

tfamily darkMatterProfileClass

tfamily void calculationReset() Reset the calculation state of the object.

tfamily double precision circularVelocity(<double precision> radius→) Returns the circular velocity (in km/s) in the dark matter profile of node at the given radius (given in units of Mpc).

tfamily double precision circularVelocityMaximum() Returns the maximum circular velocity (in km/s) in the dark matter profile of node.

tfamily double precision density(<double precision> radius→) Returns the density (in \(M_\odot\ Mpc^{-3}\)) in the dark matter profile of node at the given radius (given in units of Mpc).

tfamily double precision enclosedMass(<double precision> radius→) Returns the enclosed mass (in \(M_\odot\)) in the dark matter profile of node at the given radius (given in units of Mpc) for the given node.

tfamily double precision energy() Returns the total energy for the given node in units of \(M_\odot\ km^2 s^{-1}\).

tfamily double precision energyGrowthRate() Returns the rate of change of the total energy of node in units of \(M_\odot\ km^2 s^{-1}\ Gyr^{-1}\).

tfamily double precision freefallRadius(<double precision> time→) Returns the freefall radius (in Mpc) corresponding to the given time (in Gyr) in node.

tfamily double precision freeFallRadiusIncreaseRate(<double precision> time→) Returns the rate of increase of the freefall radius (in Mpc/Gyr) corresponding to the given time (in Gyr) in node.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision kSpace(<double precision> wavenumber→) Returns the normalized Fourier space density profile of the dark matter profile of node at the given wavenumber (given in units of Mpc^{-1}).

tfamily double precision potential(<double precision> radius→,<integer> status←) Returns the gravitational potential (in \((km/s)^2\)) in the dark matter profile of node at the given radius (given in units of Mpc).

tfamily double precision radiusFromSpecificAngularMomentum(<double precision> specificAngularMomentum→) Returns the radius (in Mpc) in the dark matter profile of node at which the specific angular momentum of a circular orbit equals specificAngularMomentum (specified in units of km s^{-1} Mpc).
Galacticus

tfamily double precision rotationNormalization() Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for the given node. Specifically, the normalization, \( A \), returned is such that \( V_{rot} = A J / M \)

tfamily void stateRestore(<integer> stateFile→) Restore the state of the object to file.

tfamily void stateStore(<integer> stateFile→) Store the state of the object to file.

tfamily darkMatterProfileConcentrationClass
tfamily double precision concentration() Returns the concentration parameter for the given node.

tfamily class(darkMatterProfileClass) darkMatterProfileDefinition() Returns a darkMatterProfile object describing the dark matter density profile used to define this concentration.

tfamily class(virialDensityContrastClass) densityContrastDefinition() Returns a virialDensityContrast object describing the virial density contrast used to define this concentration.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily darkMatterProfileConcentrationDiemerKravtsov2014
tfamily double precision concentration() Returns the concentration parameter for the given node.

tfamily class(darkMatterProfileClass) darkMatterProfileDefinition() Returns a darkMatterProfile object describing the dark matter density profile used to define this concentration.

tfamily class(virialDensityContrastClass) densityContrastDefinition() Returns a virialDensityContrast object describing the virial density contrast used to define this concentration.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily darkMatterProfileConcentrationDuttonMaccio2014
tfamily double precision concentration() Returns the concentration parameter for the given node.

tfamily class(darkMatterProfileClass) darkMatterProfileDefinition() Returns a darkMatterProfile object describing the dark matter density profile used to define this concentration.

tfamily class(virialDensityContrastClass) densityContrastDefinition() Returns a virialDensityContrast object describing the virial density contrast used to define this concentration.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily darkMatterProfileConcentrationGao2008
tfamily double precision concentration() Returns the concentration parameter for the given node.

tfamily class(darkMatterProfileClass) darkMatterProfileDefinition() Returns a darkMatterProfile object describing the dark matter density profile used to define this concentration.

tfamily class(virialDensityContrastClass) densityContrastDefinition() Returns a virialDensityContrast object describing the virial density contrast used to define this concentration.

tfamily logical isFinalizable() Return true if this object can be finalized.
16.5. Objects

tfamily darkMatterProfileConcentrationMunozCuartas2011
tfamily double precision concentration() Returns the concentration parameter for the given node.
tfamily class(darkMatterProfileClass) darkMatterProfileDefinition() Returns a darkMatterProfile object describing the dark matter density profile used to define this concentration.
tfamily class(virialDensityContrastClass) densityContrastDefinition() Returns a virialDensityContrast object describing the virial density contrast used to define this concentration.
tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily darkMatterProfileConcentrationNFW1996
tfamily double precision concentration() Returns the concentration parameter for the given node.
tfamily class(darkMatterProfileClass) darkMatterProfileDefinition() Returns a darkMatterProfile object describing the dark matter density profile used to define this concentration.
tfamily class(virialDensityContrastClass) densityContrastDefinition() Returns a virialDensityContrast object describing the virial density contrast used to define this concentration.
tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily darkMatterProfileConcentrationPrada2011
tfamily double precision concentration() Returns the concentration parameter for the given node.
tfamily class(darkMatterProfileClass) darkMatterProfileDefinition() Returns a darkMatterProfile object describing the dark matter density profile used to define this concentration.
tfamily class(virialDensityContrastClass) densityContrastDefinition() Returns a virialDensityContrast object describing the virial density contrast used to define this concentration.
tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily darkMatterProfileConcentrationWDM
tfamily double precision concentration() Returns the concentration parameter for the given node.
tfamily class(darkMatterProfileClass) darkMatterProfileDefinition() Returns a darkMatterProfile object describing the dark matter density profile used to define this concentration.
tfamily class(virialDensityContrastClass) densityContrastDefinition() Returns a virialDensityContrast object describing the virial density contrast used to define this concentration.
tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily darkMatterProfileConcentrationZhao2009
tfamily double precision concentration() Returns the concentration parameter for the given node.
tfamily class(darkMatterProfileClass) darkMatterProfileDefinition() Returns a darkMatterProfile object describing the dark matter density profile used to define this concentration.
tfamily class(virialDensityContrastClass) densityContrastDefinition() Returns a virialDensityContrast object describing the virial density contrast used to define this concentration.
tfamily logical isFinalizable() Return true if this object can be finalized.
16. Coding GALACTICUS

tfamily darkMatterProfileEinasto
tfamily void calculationReset() Reset the calculation state of the object.
tfamily double precision circularVelocity(<double precision> radius→) Returns the circular velocity (in km/s) in the dark matter profile of node at the given radius (given in units of Mpc).
tfamily double precision circularVelocityMaximum() Returns the maximum circular velocity (in km/s) in the dark matter profile of node.
tfamily double precision density(<double precision> radius→) Returns the density (in $M_\odot$ Mpc$^{-3}$) in the dark matter profile of node at the given radius (given in units of Mpc).
tfamily doublezero densityScaleFree(<double> radius→, <double> concentration→, <double> alpha→) Returns the density (in units such that the virial mass and scale length are unity) in an Einasto dark matter profile with given concentration and alpha at the given radius (given in units of the scale radius).
tfamily double precision enclosedMass(<double precision> radius→) Returns the enclosed mass (in $M_\odot$) in the dark matter profile of node at the given radius (given in units of Mpc) for the given node.
tfamily doublezero enclosedMassScaleFree(<double> radius→, <double> concentration→, <double> alpha→) Returns the enclosed mass (in units of the virial mass) in an Einasto dark matter profile with given concentration at the given radius (given in units of the scale radius).
tfamily double precision energy() Return the total energy for the given node in units of $M_\odot$ km$^2$ s$^{-1}$.
tfamily double precision energyGrowthRate() Returns the rate of chance of the total energy of node in units of $M_\odot$ km$^2$ s$^{-1}$ Gyr$^{-1}$.
tfamily void energyTableMake(<double> concentrationRequired→, <double> alphaRequired→) Create a tabulation of the energy of Einasto profiles as a function of their concentration of $\alpha$ parameter.
tfamily void fourierProfileTableMake(<double> wavenumberRequired→, <double> concentrationRequired→, <double> alphaRequired→) Create a tabulation of the Fourier transform of Einasto profiles as a function of their $\alpha$ parameter and dimensionless wavenumber.
tfamily double precision freefallRadius(<double precision> time→) Returns the freefall radius (in Mpc) corresponding to the given time (in Gyr) in node.
tfamily double precision freeFallRadiusIncreaseRate(<double precision> time→) Returns the rate of increase of the freefall radius (in Mpc/Gyr) corresponding to the given time (in Gyr) in node.
tfamily void freefallTabulate(<double> freefallTimeScaleFree→, <double> alphaRequired→) Tabulates the freefall time vs. freefall radius for Einasto halos.
tfamily doublezero freefallTimeScaleFree(<double> radius→, <double> alpha→) Compute the freefall time in a scale-free Einasto halo.
tfamily logical isFinalizable() Return true if this object can be finalized.
tfamily double precision kSpace(<double precision> wavenumber→) Returns the normalized Fourier space density profile of the dark matter profile of node at the given wavenumber (given in units of Mpc$^{-1}$).
16.5. Objects

tfamily double precision potential(<double precision> radius→, <integer> status←) Returns the gravitational potential (in \((\text{km/s})^2\)) in the dark matter profile of node at the given radius (given in units of Mpc).

tfamily \doublezero potentialScaleFree(<double> radius→, <double> concentration→, <double> alpha→) Returns the gravitational potential (in units where the virial mass and scale radius are unity) in an Einasto dark matter profile with given concentration and alpha at the given radius (given in units of the scale radius).

tfamily double precision radiusFromSpecificAngularMomentum(<double precision> specificAngularMomentum→) Returns the radius (in Mpc) in the dark matter profile of node at which the specific angular momentum of a circular orbit equals specificAngularMomentum (specified in units of km s\(^{-1}\) Mpc.

tfamily \doublezero radiusFromSpecificAngularMomentumScaleFree(<double> alpha→, <double> specificAngularMomentumScaleFree→) Compute the radius at which a circular orbit has the given specificAngularMomentumScaleFree in a scale free Einasto profile.

tfamily \void radiusFromSpecificAngularMomentumTableMake(<double> alphaRequired→, <double> specificAngularMomentumRequired→) Create a tabulation of the relation between specific angular momentum and radius in an Einasto profile.

tfamily double precision rotationNormalization() Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for the given node. Specifically, the normalization, \(A\), returned is such that \(V_{rot} = AJ/M\)

tfamily void stateRestore(<integer> stateFile→) Restore the state of the object to file.

tfamily void stateStore(<integer> stateFile→) Store the state of the object to file.

tfamily darkMatterProfileIsothermal

tfamily void calculationReset() Reset the calculation state of the object.

tfamily double precision circularVelocity(<double precision> radius→) Returns the circular velocity (in km/s) in the dark matter profile of node at the given radius (given in units of Mpc).

tfamily double precision circularVelocityMaximum() Returns the maximum circular velocity (in km/s) in the dark matter profile of node.

tfamily double precision density(<double precision> radius→) Returns the density (in \(\text{M}_\odot \text{Mpc}^{-3}\)) in the dark matter profile of node at the given radius (given in units of Mpc).

tfamily double precision enclosedMass(<double precision> radius→) Returns the enclosed mass (in \(\text{M}_\odot\)) in the dark matter profile of node at the given radius (given in units of Mpc). for the given node.

tfamily double precision energy() Return the total energy for the given node in units of \(\text{M}_\odot \text{km}^2 \text{s}^{-1}\).

tfamily double precision energyGrowthRate() Returns the rate of chance of the total energy of node in units of \(\text{M}_\odot \text{km}^2 \text{s}^{-1} \text{Gyr}^{-1}\).

tfamily double precision freefallRadius(<double precision> time→) Returns the freefall radius (in Mpc) corresponding to the given time (in Gyr) in node.

tfamily double precision freeFallRadiusIncreaseRate(<double precision> time→) Returns the rate of increase of the freefall radius (in Mpc/Gyr) corresponding to the given time (in Gyr) in node.
16. **Coding GALACTICUS**

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision kSpace(<double precision> wavenumber→) Returns the normalized Fourier space density profile of the dark matter profile of node at the given waveNumber (given in units of Mpc⁻¹).

tfamily double precision potential(<double precision> radius→,<integer> status←) Returns the gravitational potential (in (km/s)²) in the dark matter profile of node at the given radius (given in units of Mpc).

tfamily double precision radiusFromSpecificAngularMomentum(<double precision> specificAngularMomentum→) Returns the radius (in Mpc) in the dark matter profile of node at which the specific angular momentum of a circular orbit equals specificAngularMomentum (specified in units of km s⁻¹ Mpc).

tfamily double precision rotationNormalization() Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for the given node. Specifically, the normalization, A, returned is such that \( V_{\text{rot}} = AJ/M \)

tfamily void stateRestore(<integer> stateFile→) Restore the state of the object to file.

tfamily void stateStore(<integer> stateFile→) Store the state of the object to file.

tfamily darkMatterProfileNFW
tfamily \( \text{doublezero angularMomentumScaleFree(<double> concentration→}) \) Returns the total angular momentum in an NFW dark matter profile with given concentration.

tfamily void calculationReset() Reset the calculation state of the object.

tfamily double precision circularVelocity(<double precision> radius→) Returns the circular velocity (in km/s) in the dark matter profile of node at the given radius (given in units of Mpc).

tfamily double precision circularVelocityMaximum() Returns the maximum circular velocity (in km/s) in the dark matter profile of node.

tfamily double precision density(<double precision> radius→) Returns the density (in \( M_\odot \) Mpc⁻³) in the dark matter profile of node at the given radius (given in units of Mpc).

tfamily \( \text{doublezero densityScaleFree(<double> radius→, <double> concentration→, <double> alpha→}) \) Returns the density (in units such that the virial mass and scale length are unity) in an NFW dark matter profile with given concentration and alpha at the given radius (given in units of the scale radius).

tfamily double precision enclosedMass(<double precision> radius→) Returns the enclosed mass (in \( M_\odot \)) in the dark matter profile of node at the given radius (given in units of Mpc), for the given node.

tfamily \( \text{doublezero enclosedMassScaleFree(<double> radius→, <double> concentration→, <double> alpha→}) \) Returns the enclosed mass (in units of the virial mass) in an NFW dark matter profile with given concentration at the given radius (given in units of the scale radius).

tfamily double precision energy() Return the total energy for the given node in units of \( M_\odot \) km² s⁻¹.

tfamily double precision energyGrowthRate() Returns the rate of change of the total energy of node in units of \( M_\odot \) km² s⁻¹ Gyr⁻¹.
tfamily double precision freefallRadius(<double precision> time) Returns the freefall radius (in Mpc) corresponding to the given time (in Gyr) in node.

tfamily double precision freeFallRadiusIncreaseRate(<double precision> time) Returns the rate of increase of the freefall radius (in Mpc/Gyr) corresponding to the given time (in Gyr) in node.

tfamily void freefallTabulate(<double> freefallTimeScaleFree, <double> alphaRequired) Tabulates the freefall time vs. freefall radius for NFW halos.

tfamily doublezero freefallTimeScaleFree(<double> radius, <double> alpha) Compute the freefall time in a scale-free NFW halo.

tfamily void inverseAngularMomentum(<double> specificAngularMomentum) Tabulates the specific angular momentum vs. radius in an NFW profile for rapid inversion.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision kSpace(<double precision> wavenumber) Returns the normalized Fourier space density profile of the dark matter profile of node at the given wavenumber (given in units of Mpc\(^{-1}\)).

tfamily double precision potential(<double precision> radius, <integer> status) Returns the gravitational potential (in \((\text{km/s})^2\)) in the dark matter profile of node at the given radius (given in units of Mpc).

tfamily doublezero profileEnergy(<double> concentration) Computes the total energy of an NFW profile halo of given concentration.

tfamily double precision radiusFromSpecificAngularMomentum(<double precision> specificAngularMomentum) Returns the radius (in Mpc) in the dark matter profile of node at which the specific angular momentum of a circular orbit equals specificAngularMomentum (specified in units of \(\text{km s}^{-1}\) Mpc).

tfamily void stateRestore(<integer> stateFile) Restore the state of the object to file.

tfamily void stateStore(<integer> stateFile) Store the state of the object to file.

tfamily void tabulate(<double> concentration) Tabulate properties of the NFW halo profile which must be computed numerically.

tfamily hashPerfect

tfamily void create(<integer(kind=kind_int8)(:)> keys, <integer(kind=kind_int8)(:)> [values], <logical>[keepInverseTable]) Create a perfect hash.

tfamily void destroy() Destroy a perfect hash.
16. Coding GALACTICUS

tfamily \textcolor{red}{\textless integer(kind=kind_int8)\textgreater} index(<integer(kind=kind_int8)> key→) Return the index corresponding to a key in a perfect hash.

tfamily \textcolor{red}{\textless integer(kind=kind_int8)\textgreater} isPresent(<integer(kind=kind_int8)> key→) Test if a key is present in a perfect hash.

tfamily \textcolor{red}{\textless integer(kind=kind_int8)\textgreater} size() Return the size of a perfect hash.

tfamily \textcolor{red}{\textless integer(kind=kind_int8)\textgreater} value(<integer(kind=kind_int8)> key→) Return the value corresponding to a key in a perfect hash.

tfamily hdf5Object

tfamily \textcolor{red}{\textless character(len=\*)\textgreater} destroy() Destroy an HDF5 object.

tfamily \textcolor{red}{\textless character(len=\*)\textgreater} hasAttribute(<character(len=\*)> [attributeName]→) Check if an object has a named attribute.

tfamily \textcolor{red}{\textless character(len=\*)\textgreater} hasDataset(<character(len=\*)> [datasetName]→) Check if an object has a named dataset.

tfamily \textcolor{red}{\textless character(len=\*)\textgreater} hasGroup(<character(len=\*)> [groupName]→) Check if an object has a named group.

tfamily \textcolor{red}{\textless character(len=\*)\textgreater} isOpen() Return true if an object is open.

tfamily \textcolor{red}{\textless character(len=\*)\textgreater} isReference() Return true if a dataset is a reference.
16.5. Objects

```
tfamily\textcolor{red}{\textless type(hdf5Object)}\textgreater\{\textless character(len=\*)\}\textgreater\textbackslash pathTo()\ Returns the path to a given object.
```

```
tfamily\intzero rank()\ Return the rank of a dataset.
```

```
tfamily\void readAttribute\{\textless character(len=\*)\}\textgreater\textbackslash attributeName\rightarrow,\\textless(character|integer(kind=\textlsq int8)|double|character(len=\*)|type(varying_string))\[[:\]\textlsq 0\to 1\]\textgreater\textbackslash attributeValue\leftarrow,\\textless logical\textgreater\allowPseudoScalar\rightarrow\ Read an attribute from an HDF5 object.
```

```
tfamily\void readAttributeStatic\{\textless character(len=\*)\}\textgreater\textbackslash attributeName\rightarrow,\\textless(character|integer(kind=\textlsq int8)|double|character(len=\*)|type(varying_string))\[[:\]\textlsq 0\to 1\]\textgreater\textbackslash attributeValue\leftarrow,\\textless logical\textgreater\allowPseudoScalar\rightarrow\ Read an attribute from an HDF5 object into a static array.
```

```
tfamily\void readDataset\{\textless character(len=\*)\}\textgreater\textbackslash datasetName\rightarrow,\\textless(character|integer(kind=\textlsq int8)|double|character(len=\*)|type(varying_string))\[[:\]\textlsq 0\to 1\]\textgreater\textbackslash datasetValue\leftarrow,\\textless integer(kind=HSIZE_T)\[1\]\textlsq readBegin\rightarrow,\\textless integer(kind=HSIZE_T)\[1\]\textlsq readCount\rightarrow\ Read a dataset from an HDF5 group into an allocatable array.
```

```
tfamily\void readDatasetStatic\{\textless character(len=\*)\}\textgreater\textbackslash datasetName\rightarrow,\\textless(character|integer(kind=\textlsq int8)|double|character(len=\*)|type(varying_string))\[[:\]\textlsq 0\to 1\]\textgreater\textbackslash datasetValue\leftarrow,\\textless integer(kind=HSIZE_T)\[1\]\textlsq readBegin\rightarrow,\\textless integer(kind=HSIZE_T)\[1\]\textlsq readCount\rightarrow\ Read a dataset from an HDF5 group into a static array.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
tfamily\\textcolor{red}{\textless integer(kind=HSIZE_T)}\textgreater\size\{\textless integer\}\textgreater\textbackslash dim\rightarrow\ Return the size of a dataset.
```

```
2For double datasets, up to 5-dimensional datasets are supported.
```
tfamily history

tfamily \textcolor{red}{\textless type(history)\textgreater} add(<type(history)> + <type(history)>)
Addition operator.

tfamily \textcolor{red}{\textless type(node)\textgreater} historyDefinition \rightarrow Build a history object from an XML definition.

tfamily \textcolor{red}{\textlesstype(history)\textgreater} divide(<type(history)>/<type(history)>)
Division operator.

tfamily \textcolor{red}{\textlesstype(history)\textgreater} multiply(<type(history)>*<type(history)>)
Multiplication operator.

tfamily \textcolor{red}{\textlesstype(history)\textgreater} subtract(<type(history)>-<type(history)>)
Subtraction operator.

tfamily \textcolor{red}{\textlesstype(history)\textgreater} divide(<type(history)>/<type(history)>)
Division operator.

tfamily \textcolor{red}{\textlesstype(history)\textgreater} multiply(<type(history)>*<type(history)>)
Multiplication operator.

tfamily \textcolor{red}{\textlesstype(history)\textgreater} subtract(<type(history)>-<type(history)>)
Subtraction operator.
tfamily hotHaloMassDistributionBetaProfile

tfamily double precision density(<double precision> radius→) Return the density of the hot halo at the given radius.

tfamily double precision densityLogSlope(<double precision> radius→) Return the logarithmic slope of the density of the hot halo at the given radius.

tfamily double precision enclosedMass(<double precision> radius→) Return the mass enclosed in the hot halo at the given radius.

tfamily void initialize(<type(treeNode)> node→) Initialize the \( \beta \)-profile density hot halo mass distribution for the given node.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision radialMoment(<double precision> moment→,<double precision> radius→) Return the density of the hot halo at the given radius.

tfamily double precision rotationNormalization() Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for node. Specifically, the normalization, \( A \), returned is such that \( V_{\text{rot}} = AJ/M \).

---

363
tfamily hotHaloMassDistributionCoreRadiusGrowing
   tfamily logical isFinalizable() Return true if this object can be finalized.
   tfamily double precision radius() Return the core radius of the hot halo mass distribution.
   tfamily void stateRestore(<integer> stateFile→) Restore the state of the object to file.
   tfamily void stateStore(<integer> stateFile→) Store the state of the object to file.

tfamily hotHaloMassDistributionCoreRadiusVirialFraction
   tfamily logical isFinalizable() Return true if this object can be finalized.
   tfamily double precision radius() Return the core radius of the hot halo mass distribution.
   tfamily void stateRestore(<integer> stateFile→) Restore the state of the object to file.
   tfamily void stateStore(<integer> stateFile→) Store the state of the object to file.

tfamily hotHaloMassDistributionNull
   tfamily double precision density(<double precision> radius→) Return the density of the hot halo at the given radius.
   tfamily double precision densityLogSlope(<double precision> radius→) Return the logarithmic slope of the density of the hot halo at the given radius.
   tfamily double precision enclosedMass(<double precision> radius→) Return the mass enclosed in the hot halo at the given radius.
   tfamily logical isFinalizable() Return true if this object can be finalized.
   tfamily double precision radialMoment(<double precision> moment→,<double precision> radius→) Return the density of the hot halo at the given radius.
   tfamily double precision rotationNormalization() Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for node. Specifically, the normalization, $A$, returned is such that $V_{rot} = AJ/M$.

tfamily hotHaloMassDistributionRicotti2000
   tfamily double precision density(<double precision> radius→) Return the density of the hot halo at the given radius.
   tfamily double precision densityLogSlope(<double precision> radius→) Return the logarithmic slope of the density of the hot halo at the given radius.
   tfamily double precision enclosedMass(<double precision> radius→) Return the mass enclosed in the hot halo at the given radius.
   tfamily void initialize(<*type(treeNode)> node→) Initialize the $\beta$-profile density hot halo mass distribution for the given node.
   tfamily logical isFinalizable() Return true if this object can be finalized.
16.5. Objects

tfamily double precision radialMoment(<double precision> moment→,<double precision> radius→)
Return the density of the hot halo at the given radius.

tfamily double precision rotationNormalization() Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for node. Specifically, the normalization, \( A \), returned is such that \( V_{rot} = AJ/M \).

tfamily importerUnits

tfamily \textcolor{red}{\textless type(importerUnits)\textgreater} exponentiate(<integer> exponent→)
Raise to the given integer power.

tfamily \textcolor{red}{\textless type(importerUnits)\textgreater} multiply(<type(importerUnits)> units2→)
Multiply by another importerUnits object.

tfamily inputParameterList

tfamily \void add(<character(len=*>) name→, <character(len=*>) value→) Add a parameter and value to the list.

tfamily \void outputToXML() Write a list fo input parameters to an XML document.

tfamily integerScalarHash

tfamily \void delete(<(character(len=*)|varying_string)> key→, <integer> value→) Delete a key from the hash.

tfamily void destroy() Destroy the hash.

tfamily \logicalzero exists(<(character(len=*)|varying_string|<integer>)> key→) Return true if the specified key exists in the hash.

tfamily \void initialize() Initialize the hash.

tfamily \textcolor{red}{\textless type(varying_string)\[:\]} key(<integer> indexValue→) Return the key of the indexValue\(^{th}\) entry in the hash.

tfamily \void keys(<type(varying_string)[:]> keys↔) Return an array of all keys in the hash.

tfamily \void set(<(character(len=*)|varying_string)> key→, <integer> value→) Set the value of a key in the hash.

tfamily \intzero size() Return the number of keys in the hash.

tfamily \intzero value(<(character(len=*)|varying_string|<integer>)> key→, <integer> value→) Return the value for the given key.

tfamily \void values(<integer[:]> values↔) Return an array of all values in the hash.
tfamily intergalacticMediumStateClass

   tfamily double precision doublyIonizedHeliumFraction(<double precision> time→) Return the doubly-ionized fraction of helium in the IGM at the given time.

   tfamily double precision electronFraction(<double precision> time→) Return the electron fraction (relative to hydrogen) in the IGM at the given time.

   tfamily double precision electronScatteringOpticalDepth(<double precision> time→,<logical> assume-FullyIonized→) Return the electron scattering optical depth from the present day back to the given time in the IGM.

   tfamily double precision electronScatteringTime(<double precision> opticalDepth→,<logical> assume-FullyIonized→) Return the cosmological time at which the given electron scattering opticalDepth is reached (integrating from the present day) in the IGM.

   tfamily logical isFinalizable() Return true if this object can be finalized.

   tfamily double precision neutralHeliumFraction(<double precision> time→) Return the neutral fraction of helium in the IGM at the given time.

   tfamily double precision neutralHydrogenFraction(<double precision> time→) Return the neutral fraction of hydrogen in the IGM at the given time.

   tfamily double precision singlyIonizedHeliumFraction(<double precision> time→) Return the singly-ionized fraction of helium in the IGM at the given time.

   tfamily double precision singlyIonizedHydrogenFraction(<double precision> time→) Return the singly-ionized fraction of hydrogen in the IGM at the given time.

   tfamily double precision temperature(<double precision> time→) Return the temperature (in Kelvin) of the IGM at the given time.

tfamily intergalacticMediumStateFile

   tfamily double precision doublyIonizedHeliumFraction(<double precision> time→) Return the doubly-ionized fraction of helium in the IGM at the given time.

   tfamily double precision electronFraction(<double precision> time→) Return the electron fraction (relative to hydrogen) in the IGM at the given time.

   tfamily double precision electronScatteringOpticalDepth(<double precision> time→,<logical> assume-FullyIonized→) Return the electron scattering optical depth from the present day back to the given time in the IGM.

   tfamily double precision electronScatteringTime(<double precision> opticalDepth→,<logical> assume-FullyIonized→) Return the cosmological time at which the given electron scattering opticalDepth is reached (integrating from the present day) in the IGM.

   tfamily logical isFinalizable() Return true if this object can be finalized.

   tfamily double precision neutralHeliumFraction(<double precision> time→) Return the neutral fraction of helium in the IGM at the given time.

   tfamily double precision neutralHydrogenFraction(<double precision> time→) Return the neutral fraction of hydrogen in the IGM at the given time.
16.5. Objects

tfamily double precision singlyIonizedHeliumFraction(<double precision> time→) Return the singly-ionized fraction of helium in the IGM at the given time.

tfamily double precision singlyIonizedHydrogenFraction(<double precision> time→) Return the singly-ionized fraction of hydrogen in the IGM at the given time.

tfamily double precision temperature(<double precision> time→) Return the temperature (in Kelvin) of the IGM at the given time.

tfamily intergalacticMediumStateInternal
tfamily \void densityH1Set(<double(:)> densityHydrogen1→) Set the density of neutral hydrogen time series.

tfamily \void densityH2Set(<double(:)> densityHydrogen2→) Set the density of ionized hydrogen time series.

tfamily \void densityHe1Set(<double(:)> densityHelium1→) Set the density of neutral helium time series.

tfamily \void densityHe2Set(<double(:)> densityHelium2→) Set the density of singly-ionized helium time series.

tfamily \void densityHe3Set(<double(:)> densityHelium3→) Set the density of doubly-ionized helium time series.

tfamily double precision doublyIonizedHeliumFraction(<double precision> time→) Return the doubly-ionized fraction of helium in the IGM at the given time.

tfamily double precision electronFraction(<double precision> time→) Return the electron fraction (relative to hydrogen) in the IGM at the given time.

tfamily double precision electronScatteringOpticalDepth(<double precision> time→,<logical> assumeFullyIonized→) Return the electron scattering optical depth from the present day back to the given time in the IGM.

tfamily double precision electronScatteringTime(<double precision> opticalDepth→,<logical> assumeFullyIonized→) Return the cosmological time at which the given electron scattering opticalDepth is reached (integrating from the present day) in the IGM.

tfamily \double zero filteringMAss(<double> time→) Return the filtering mass at the given time.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily \void massFilteringSet(<double(:)> massFiltering→) Set the filtering mass time series.

tfamily double precision neutralHeliumFraction(<double precision> time→) Return the neutral fraction of helium in the IGM at the given time.

tfamily double precision neutralHydrogenFraction(<double precision> time→) Return the neutral fraction of hydrogen in the IGM at the given time.

tfamily double precision singlyIonizedHeliumFraction(<double precision> time→) Return the singly-ionized fraction of helium in the IGM at the given time.
16. Coding GALACTICUS

tfamily double precision singlyIonizedHydrogenFraction(<double precision> time→) Return the singly-ionized fraction of hydrogen in the IGM at the given time.

tfamily double precision temperature(<double precision> time→) Return the temperature (in Kelvin) of the IGM at the given time.

tfamily void temperatureSet(<double: > temperature→) Set the temperature time series.

tfamily void timeSet(<double: > times→) Set the times to use for all time series.

tfamily intergalacticMediumStateRecFast

tfamily double precision doublyIonizedHeliumFraction(<double precision> time→) Return the doubly-ionized fraction of helium in the IGM at the given time.

tfamily double precision electronFraction(<double precision> time→) Return the electron fraction (relative to hydrogen) in the IGM at the given time.

tfamily double precision electronScatteringOpticalDepth(<double precision> time→, <logical> assumeFullyIonized→) Return the electron scattering optical depth from the present day back to the given time in the IGM.

tfamily double precision electronScatteringTime(<double precision> opticalDepth→, <logical> assumeFullyIonized→) Return the cosmological time at which the given electron scattering opticalDepth is reached (integrating from the present day) in the IGM.

ntfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision neutralHeliumFraction(<double precision> time→) Return the neutral fraction of helium in the IGM at the given time.

tfamily double precision neutralHydrogenFraction(<double precision> time→) Return the neutral fraction of hydrogen in the IGM at the given time.

tfamily double precision singlyIonizedHeliumFraction(<double precision> time→) Return the singly-ionized fraction of helium in the IGM at the given time.

tfamily double precision singlyIonizedHydrogenFraction(<double precision> time→) Return the singly-ionized fraction of hydrogen in the IGM at the given time.

tfamily double precision temperature(<double precision> time→) Return the temperature (in Kelvin) of the IGM at the given time.

tfamily intergalacticMediumStateSimple

tfamily double precision doublyIonizedHeliumFraction(<double precision> time→) Return the doubly-ionized fraction of helium in the IGM at the given time.

tfamily double precision electronFraction(<double precision> time→) Return the electron fraction (relative to hydrogen) in the IGM at the given time.

tfamily double precision electronScatteringOpticalDepth(<double precision> time→, <logical> assumeFullyIonized→) Return the electron scattering optical depth from the present day back to the given time in the IGM.
tfamily double precision electronScatteringTime(<double precision> opticalDepth→,<logical> assumeFullyIonized→) Return the cosmological time at which the given electron scattering opticalDepth is reached (integrating from the present day) in the IGM.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily double precision neutralHeliumFraction(<double precision> time→) Return the neutral fraction of helium in the IGM at the given time.

tfamily double precision neutralHydrogenFraction(<double precision> time→) Return the neutral fraction of hydrogen in the IGM at the given time.

tfamily double precision singlyIonizedHeliumFraction(<double precision> time→) Return the singly-ionized fraction of helium in the IGM at the given time.

tfamily double precision singlyIonizedHydrogenFraction(<double precision> time→) Return the singly-ionized fraction of hydrogen in the IGM at the given time.

tfamily double precision temperature(<double precision> time→) Return the temperature (in Kelvin) of the IGM at the given time.

tfamily keplerOrbit

tfamily doublezero angularMomentum() Returns the angular momentum of an orbit.

tfamily void angularMomentumSet(<double> angularMomentum→) Sets the angular momentum of an orbit.

tfamily void assertIsDefined() Asserts that an orbit is fully defined.

tfamily void builder(<*type(node)> keplerOrbitDefinition→) Build a Kepler orbit from an XML definition.

tfamily void destroy() Destroys an orbit.

tfamily void dump() Dump an orbit.

tfamily void dumpRaw(<integer> fileHandle→) Dump an orbit in binary.

tfamily doublezero eccentricity() Returns the eccentricity of an orbit.

tfamily void eccentricitySet(<double> eccentricity→) Sets the eccentricity of an orbit.

tfamily doublezero energy() Returns the energy of an orbit.

tfamily void energySet(<double> energy→) Sets the energy of an orbit.

tfamily doublezero hostMass() Returns the host mass of an orbit.

tfamily logical zero isBound() Returns true if the orbit is bound.

tfamily logical zero isDefined() Returns true if an orbit is fully defined.

tfamily void massesSet(<double> satelliteMass→, <double> hostMass→) Sets the masses of satellite and host objects.
tfamily \ void output(<integer> integerProperty\leftrightarrow, <integer> integerBufferCount\leftrightarrow, <integer(:,:)> integerBuffer\leftrightarrow, <integer> doubleProperty\leftrightarrow, <integer> doubleBufferCount\leftrightarrow, <double(:,:)>
doubleBuffer\leftrightarrow, <double> time→, <integer> instance→) Store a keplerOrbit object in the output buffers.

tfamily \ void outputCount(<integer> integerPropertyCount\leftrightarrow, <integer> doublePropertyCount\leftrightarrow, <double>
time→, <integer> instance→) Specify the count of a keplerOrbit object for output.

tfamily \ void outputNames(<integer> integerProperty\leftrightarrow, <char[*](::)> integerPropertyNames\leftrightarrow, <char[*](::)> integerPropertyComments\leftrightarrow, <double(:,:)> integerPropertyUnitsSI\leftrightarrow, <integer> doubleProperty\leftrightarrow, <char[*](::)> doublePropertyNames\leftrightarrow, <char[*](::)> doublePropertyComments\leftrightarrow, <double(:,:)>
doublePropertyUnitsSI\leftrightarrow, <double> time→, <integer> instance→) Specify the names of a keplerOrbit object properties for output.

tfamily \ void propagate(<double> velocityRadial→) Propagates an orbit to a new position.

tamily \ doublezero radius() Returns the radius of an orbit.

tfamily \ doublezero radiusApocenter() Returns the apocenter radius of an orbit.

tamily \ void radiusApocenterSet(<double> radius→) Sets the apocenter radius of an orbit.

tfamily \ doublezero radiusPericenter() Returns the pericenter radius of an orbit.

tamily \ void radiusPericenterSet(<double> radius→) Sets the pericenter radius of an orbit.

tamily \ void radiusSet(<double> radius→) Sets the radius of an orbit.

tamily \ void readRaw(<integer> fileHandle→) Read an orbit in binary.

tamily \ void reset() Resets an orbit to a null state.

tamily \ doublezero semiMajorAxis() Returns the semi-major axis of an orbit.

tamily \ void semiMajorAxisSet(<double> semiMajorAxis→) Sets the semi-major axis of an orbit.

tamily \ doublezero specificReducedMass() Returns the specific reduced mass (i.e. the reduced mass per unit satellite mass, \( \mu_s = \frac{M_{\text{host}}}{(M_{\text{satellite}} + M_{\text{host}})} \)) of the orbit.

tamily \ doublezero velocityRadial() Returns the radial velocity of an orbit.

tamily \ void velocityRadialSet(<double> newRadius→) Sets the radial velocity of an orbit.

tamily \ doublezero velocityScale() Returns the velocity scale of an orbit.

tamily \ doublezero velocityTangential() Returns the tangential velocity of an orbit.

tamily \ void velocityTangentialSet(<double> velocityTangential→) Sets the tangential velocity of an orbit.
16.5. Objects

tfamily longIntegerHistory

tfamily \void builder(<*type(node)> historyDefinition→) Build a history object from an XML definition.

tfamily \void clone(<type(history)> historyToClone→) Clone a history object.

tfamily \void create(<integer> historyCount→, <integer> timesCount→, <double> [timeBegin]→, <double> [timeEnd]→, <integer> [rangeType]→) Creates a history object with a specified range of times.

tfamily \void destroy(<type(history)> historyToClone→) Destroys a history object.

tfamily \void dump() Dump a history object.

tfamily \void dumpRaw(<integer> fileHandle→) Dump a history object in binary.

tfamily \logicalzero exists() Returns true if the given history has been created.

tfamily \void readRaw(<integer> fileHandle→) Read a history object in binary.

tfamily \void reset() Resets all entries in a history to zero.

tfamily \void trim(<double> currentTime→, <integer> [minimumPointsToRemove]→) Removes any times in a history which have become outdated.

tfamily massDistribution

tfamily \doublezero density(<class(coordinate)> coordinates→) Returns the density of the mass distribution at the supplied coordinates.

tfamily \doublezero densityGradientRadial(<class(coordinate)> coordinates→, <logical> [logarithmic]→) Returns the gradient with respect to radius of the density of the mass distribution at the supplied coordinates. If the optional logarithmic argument is set to true, return the logarithmic gradient.

tfamily \doublezero densityRadialMoment(<double> moment→, <double> radiusMinimum→, <double> radiusMaximum→, <logical> [isInfinite]←) Returns the $n^{th}$ moment of the integral of the density over radius, $\int_{0}^{\infty} \rho(x)|x|^n dx$.

tfamily \logicalzero isDimensionless() Returns true if this is a dimensionless mass distribution, false otherwise.

tfamily \doublezero massEnclosedBySphere(<double> radius→) Returns the mass enclosed by a sphere of given radius centered on the origin.

tfamily \doublezero potential(<class(coordinate)> coordinates→) Returns the gravitational potential at the specified coordinates.

tfamily \enumMassDistributionSymmetry symmetry() Returns a label specifying the symmetry of the mass distribution (see §17.5.4).
tfamily massDistributionBetaProfile

  tfamily \doublezero density(<class(coordinate)> coordinates→) Returns the density of the mass distribution at the supplied coordinates.

  tfamily \doublezero densityGradientRadial(<class(coordinate)> coordinates→, <logical> [logarithmic]→) Returns the gradient with respect to radius of the density of the mass distribution at the supplied coordinates. If the optional logarithmic argument is set to true, return the logarithmic gradient.

  tfamily \doublezero densityRadialMoment(<double> moment→, <double> radiusMinimum→, <double> radiusMaximum→, <logical> [isInfinite]←) Returns the $n^{th}$ moment of the integral of the density over radius, $\int_{0}^{\infty} \rho(x)|x|^n dx$.

  tfamily \doublezero halfMassRadius() Returns the radius enclosing half of the mass of the mass distribution.

  tfamily \void initialize(<double> beta→, <double> coreRadius→, <double> densityNormalization→, <double> mass→, <double> outerRadius→, <logical> isDimensionless→) Initialize the mass distribution.

  tfamily \logical zero isDimensionless() Returns true is this a dimensionless mass distribution, false otherwise.

  tfamily \doublezero massEnclosedBySphere(<double> radius→) Returns the mass enclosed by a sphere of given radius centered on the origin.

  tfamily \doublezero potential(<class(coordinate)> coordinates→) Returns the gravitational potential at the specified coordinates.

  tfamily \enumMassDistributionSymmetry symmetry() Returns a label specifying the symmetry of the mass distribution (see §17.5.4).

tfamily massDistributionCylindrical

  tfamily \doublezero density(<class(coordinate)> coordinates→) Returns the density of the mass distribution at the supplied coordinates.

  tfamily \doublezero densityGradientRadial(<class(coordinate)> coordinates→, <logical> [logarithmic]→) Returns the gradient with respect to radius of the density of the mass distribution at the supplied coordinates. If the optional logarithmic argument is set to true, return the logarithmic gradient.

  tfamily \doublezero densityRadialMoment(<double> moment→, <double> radiusMinimum→, <double> radiusMaximum→, <logical> [isInfinite]←) Returns the $n^{th}$ moment of the integral of the density over radius, $\int_{0}^{\infty} \rho(x)|x|^n dx$.

  tfamily \logical zero isDimensionless() Returns true is this a dimensionless mass distribution, false otherwise.

  tfamily \doublezero massEnclosedBySphere(<double> radius→) Returns the mass enclosed by a sphere of given radius centered on the origin.

  tfamily \doublezero potential(<class(coordinate)> coordinates→) Returns the gravitational potential at the specified coordinates.

  tfamily \enumMassDistributionSymmetry symmetry() Returns a label specifying the symmetry of the mass distribution (see §17.5.4).
tfamily massDistributionHernquist

tfamily \(\text{\texttt{density}}(<\texttt{coordinate}> \text{coordinates}\rightarrow)\) Returns the density of the mass distribution at the supplied coordinates.

tfamily \(\text{\texttt{densityGradientRadial}}(<\texttt{coordinate}> \text{coordinates}\rightarrow, \text{\texttt{logical}}[[\texttt{logarithmic}]\rightarrow)\)

Returns the gradient with respect to radius of the density of the mass distribution at the supplied coordinates. If the optional \texttt{logarithmic} argument is set to true, return the logarithmic gradient.

tfamily \(\text{\texttt{densityRadialMoment}}(<\texttt{double}> \text{moment}\rightarrow, <\texttt{double}> \text{radiusMinimum}\rightarrow, <\texttt{double}> \text{radiusMaximum}\rightarrow, \text{\texttt{logical}}[[\texttt{isInfinite}][\rightarrow)\)

Returns the \(n^{th}\) moment of the integral of the density over radius, \(\int_{\text{radiusMinimum}}^{\text{radiusMaximum}} \rho(x)|x^n|\text{dx}\).

tfamily \(\text{\texttt{halfMassRadius}}()\) Returns the radius enclosing half of the mass of the mass distribution.

tfamily \(\text{\texttt{initialize}}(<\texttt{double}> \text{scaleLength}\rightarrow, <\texttt{double}> \text{densityNormalization}\rightarrow, <\texttt{double}> \text{mass}\rightarrow, \text{\texttt{logical}}[[\texttt{isDimensionless}]\rightarrow)\) Initialize the mass distribution.

tfamily \(\text{\texttt{isDimensionless}}()\) Returns \texttt{true} is this is a dimensionless mass distribution, \texttt{false} otherwise.

tfamily \(\text{\texttt{massEnclosedBySphere}}(<\texttt{double}> \text{radius}\rightarrow)\) Returns the mass enclosed by a sphere of given radius centered on the origin.

tfamily \(\text{\texttt{potential}}(<\texttt{coordinate}> \text{coordinates}\rightarrow)\) Returns the gravitational potential at the specified coordinates.

tfamily \(\text{\texttt{symmetry}}()\) Returns a label specifying the symmetry of the mass distribution (see §17.5.4).

tfamily massDistributionNFW

tfamily \(\text{\texttt{density}}(<\texttt{coordinate}> \text{coordinates}\rightarrow)\) Returns the density of the mass distribution at the supplied coordinates.

tfamily \(\text{\texttt{densityGradientRadial}}(<\texttt{coordinate}> \text{coordinates}\rightarrow, \text{\texttt{logical}}[[\texttt{logarithmic}]\rightarrow)\)

Returns the gradient with respect to radius of the density of the mass distribution at the supplied coordinates. If the optional \texttt{logarithmic} argument is set to true, return the logarithmic gradient.

tfamily \(\text{\texttt{densityRadialMoment}}(<\texttt{double}> \text{moment}\rightarrow, <\texttt{double}> \text{radiusMinimum}\rightarrow, <\texttt{double}> \text{radiusMaximum}\rightarrow, \text{\texttt{logical}}[[\texttt{isInfinite}][\rightarrow)\)

Returns the \(n^{th}\) moment of the integral of the density over radius, \(\int_{\text{radiusMinimum}}^{\text{radiusMaximum}} \rho(x)|x^n|\text{dx}\).

tfamily \(\text{\texttt{halfMassRadius}}()\) Returns the radius enclosing half of the mass of the mass distribution.

tfamily \(\text{\texttt{initialize}}(<\texttt{double}> \text{scaleLength}\rightarrow, <\texttt{double}> \text{concentration}\rightarrow, <\texttt{double}> \text{densityNormalization}\rightarrow, <\texttt{double}> \text{mass}\rightarrow, <\texttt{double}> \text{virialRadius}\rightarrow, \text{\texttt{logical}}[[\texttt{isDimensionless}]\rightarrow)\) Initialize the mass distribution.

tfamily \(\text{\texttt{isDimensionless}}()\) Returns \texttt{true} is this is a dimensionless mass distribution, \texttt{false} otherwise.

tfamily \(\text{\texttt{massEnclosedBySphere}}(<\texttt{double}> \text{radius}\rightarrow)\) Returns the mass enclosed by a sphere of given radius centered on the origin.
16. Coding GALACTICUS

tfamily \doublezero potential(<\text{class(coordinate)}> coordinates→) Returns the gravitational potential at the specified coordinates.

tfamily \enumMassDistributionSymmetry symmetry() Returns a label specifying the symmetry of the mass distribution (see §17.5.4).

tfamily massDistributionSersic
tfamily \doublezero density(<\text{class(coordinate)}> coordinates→) Returns the density of the mass distribution at the supplied coordinates.

tfamily \doublezero densityGradientRadial(<\text{class(coordinate)}> coordinates→, <\text{logical}> [logarithmic]→) Returns the gradient with respect to radius of the density of the mass distribution at the supplied coordinates. If the optional logarithmic argument is set to true, return the logarithmic gradient.

tfamily \doublezero densityRadialMoment(<\text{double}>
moment→, <\text{double}>
radiusMinimum→, <\text{double}>
radiusMaximum→, <\text{logical}> [isInfinite]←) Returns the $n^{th}$ moment of the integral of the density over radius, $\int_{0}^{\infty} \rho(x)|x|^{n} dx$.

tfamily \doublezero halfMassRadius() Returns the radius enclosing half of the mass of the mass distribution.

tfamily \doublezero halfMassRadiusProjected() Return the half mass radius of the profile in projection.

tfamily \void initialize(<\text{double}>
index→, <\text{double}>
halfMassRadius→, <\text{double}>
densityNormalization→,
<\text{double}>
mass→, <\text{logical}> isDimensionless→) Initialize the mass distribution.

tfamily \logical zero isDimensionless() Returns true is this is a dimensionless mass distribution, false otherwise.

tfamily \doublezero massEnclosedBySphere(<\text{double}>
radius→) Returns the mass enclosed by a sphere of given radius centered on the origin.

tfamily \doublezero potential(<\text{class(coordinate)}> coordinates→) Returns the gravitational potential at the specified coordinates.

tfamily \enumMassDistributionSymmetry symmetry() Returns a label specifying the symmetry of the mass distribution (see §17.5.4).

tfamily massDistributionSpherical
tfamily \doublezero density(<\text{class(coordinate)}> coordinates→) Returns the density of the mass distribution at the supplied coordinates.

tfamily \doublezero densityGradientRadial(<\text{class(coordinate)}> coordinates→, <\text{logical}> [logarithmic]→) Returns the gradient with respect to radius of the density of the mass distribution at the supplied coordinates. If the optional logarithmic argument is set to true, return the logarithmic gradient.

tfamily \doublezero densityRadialMoment(<\text{double}>
moment→, <\text{double}>
radiusMinimum→, <\text{double}>
radiusMaximum→, <\text{logical}> [isInfinite]←) Returns the $n^{th}$ moment of the integral of the density over radius, $\int_{0}^{\infty} \rho(x)|x|^{n} dx$.

tfamily \doublezero halfMassRadius() Returns the radius enclosing half of the mass of the mass distribution.
16.5. Objects

tfamily `logical zero isDimensionless()` Returns `true` if this is a dimensionless mass distribution, `false` otherwise.

tfamily `double zero massEnclosedBySphere(<double> radius →)` Returns the mass enclosed by a sphere of given `radius` centered on the origin.

tfamily `double zero potential(<class(coordinate)> coordinates →)` Returns the gravitational potential at the specified `coordinates`.

tfamily `enumMassDistributionSymmetry symmetry()` Returns a label specifying the symmetry of the mass distribution (see §17.5.4).

tfamily mergerTree
tfamily `textcolor{red}{\textless *type(treeEvent)\textgreater} createEvent()` Create a `treeEvent` object in this tree.

tfamily `void destroy()` Destroys the merger tree, including all nodes and their components.

tfamily `void destroyBranch(<*type(treeNode)> thisNode ↔)` Destroys a branch of a merger tree starting from the supplied node. All nodes and their components are destroyed.

tfamily `double zero earliestTime()` Return the earliest time in a merger tree.

tfamily `double zero earliestTimeEvolving()` Return the earliest time in an evolving merger tree.

tfamily `\textcolor{red}{\textless *type(treeNode)\textgreater} getNode(<integer(kind_int8)> nodeIndex →)` Returns a pointer to the node with given index in the merger tree, or a null pointer if no such node exists.

tfamily `double zero latestTime()` Return the latest time in a merger tree.

tfamily `void removeEvent(<type(treeEvent)> event →)` Remove a `treeEvent` from this tree.

ntfamily mergerTreeData
tfamily `void addMetadata(<metaDataType> metadataType →, <character(len=*)> label →, (<>double,,<integer>,character(len=*)) value →)` Add a metadatum to the tree data structure.

tfamily `void export(<character(len=*)> outputFileName →, <character(len=*)> outputFormat →, <integer> hdfChunkSize →, <integer> hdfCompressionLevel →, <logical> [append] →)` Export the tree data to an output file.

tfamily `void makeReferences(<logical> makeReferences →)` Specify whether or not merger tree dataset references should be made.

tfamily `void nodeCountSet(<integer> nodeCount →)` Set the total number of nodes in the data structure.

tfamily `void particleCountSet(<integer> particleCount →)` Set the total number of particles in the data structure.

tfamily `void readASCII(inputFile →, [lineNumberStart] →, [lineNumberStop] →, [separator] →)` Read node data from an ASCII file into the data structure.
tfamily \void readParticlesASCII(<character(len=*>) inputFile→, <integer> [lineNumberStart]→, <integer> [lineNumberStop]→, <character(len=*)> [separator]→) Read particle data from an ASCII file into the data structure.

tfamily \void reset() Reset the data structure.

tfamily \void setDummyHostId(<integer> dummyHostId→) Set host ID for self-hosting halos if host ID is not node ID.

tfamily \void setIncludesHubbleFlow(<logical> includesHubbleFlow→) Specify if velocities include the Hubble flow.

tfamily \void setIncludesSubhaloMasses(<logical> includesSubhaloMasses→) Set whether halo masses include the masses of the subhalos.

tfamily \void setParticleMass(<double> particleMass→) Set the mass of an N-body particle in the simulation from which the trees were derived.

tfamily \void setParticlePropertyColumn(<propertyType> propertyType→, <integer> columnNumber→) Set the column in an ASCII data file corresponding to a given particle property.

tfamily \void setPositionsArePeriodic(<logical> isPeriodic→) Set if positions are periodic.

tfamily \void setProperty(<propertyType> propertyType→, <integer(kind=kind_int8)(:)|<double(:)> property→) Set a node property in the data structure.

tfamily \void setPropertyColumn(<propertyType> propertyType→, columnNumber→) Set the column in an ASCII data file corresponding to a given node property.

tfamily \void setSelfContained(<logical> areSelfContained→) Specify if trees are self-contained (i.e. contain no cross-links to other trees).

tfamily \void setUnits(<metaDataType> unitType→, <double> unitsInSI→, <integer> [hubbleExponent]→, <integer> [scaleFactorExponent]→, <character(len=*)> [name]→) Set the units used.

tfamily \void close() Closes the file.

tfamily double precision cubeLength(<double precision> time→,<integer> status←) Returns the length of the simulation cube.

tfamily void import(<integer> i→,<logical> requireScaleRadii→,<logical> requireAngularMomenta→,<logical> requireAngularMomenta3D→,<logical> requireSpin→,<logical> requireSpin3D→,<logical> requirePositions→,<logical> requireParticleCounts→,<logical> requireVelocityMaxima→,<logical> requireVelocityDispersions→) Imports the i

376
tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily logical massesIncludeSubhalos() Returns a Boolean specifying whether halo masses include the contribution from their subhalos.

tfamily integer(kind=c_size_t) nodeCount(<integer> i→) Returns the number of nodes in the \(i^{th}\) tree.

tfamily void open() Opens the file.

tfamily logical particleCountAvailable() Return true if particle counts are available.

tfamily integer positionsArePeriodic() Returns a Boolean integer specifying whether positions are periodic.

tfamily logical positionsAvailable(<logical> positions→,<logical> velocities→) Return true if positions and/or velocities are available.

tfamily logical scaleRadiiAvailable() Return true if scale radii are available.

tfamily logical spin3DAvailable() Return true if spin (vectors) are available.

tfamily logical spinAvailable() Return true if spin (magnitudes) are available.

tfamily void subhaloTrace(<double precision> time←,<double precision> position←,<double precision> velocity←) Supplies epochs, positions, and velocities for traced subhalos.

tfamily integer(kind=c_size_t) subhaloTraceCount() Returns the length of a node’s subhalo trace.

tfamily integer(kind=c_size_t) treeCount() Returns a count of the number of trees available.

tfamily integer(kind=kind_int8) treeIndex(<integer> i→) Returns the index of the \(i^{th}\) tree.

tfamily integer treesAreSelfContained() Returns a Boolean integer specifying whether trees are self-contained.

tfamily integer treesHaveSubhalos() Returns a Boolean integer specifying whether or not the trees have subhalos.

tfamily double precision treeWeight(<integer> i→) Returns the weight to assign to the \(i^{th}\) tree.

tfamily integer velocitiesIncludeHubbleFlow() Returns a Boolean integer specifying whether velocities include the Hubble flow.

tfamily logical velocityDispersionAvailable() Return true if halo velocity dispersions are available.

tfamily logical velocityMaximumAvailable() Return true if rotation curve velocity maxima are available.

tfamily mergerTreeImporterGalacticus

tfamily logical angularMomenta3DAvailable() Return true if angular momenta (vectors) are available.

tfamily logical angularMomentaAvailable() Return true if angular momenta (magnitudes) are available.

tfamily logical angularMomentaIncludeSubhalos() Returns a Boolean specifying whether halo angular momenta (or spins) include the contribution from their subhalos.

tfamily void close() Closes the file.
tfamily double precision cubeLength(<double precision> time→,<integer> status←) Returns the length of the simulation cube.

tfamily void import(<integer> i→,<logical> requireScaleRadii→,<logical> requireAngularMomenta→,<logical> requireAngularMomenta3D→,<logical> requireSpin→,<logical> requireSpin3D→,<logical> requirePositions→,<logical> requireParticleCounts→,<logical> requireVelocityMaxima→,<logical> requireVelocityDispersions→) Imports the $i^{th}$ tree.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily logical massesIncludeSubhalos() Returns a Boolean specifying whether halo masses include the contribution from their subhalos.

tfamily integer(kind=c_size_t) nodeCount(<integer> i→) Returns the number of nodes in the $i^{th}$ tree.

tfamily void open() Opens the file.

tfamily logical particleCountAvailable() Return true if particle counts are available.

tfamily integer positionsArePeriodic() Returns a Boolean integer specifying whether positions are periodic.

tfamily logical positionsAvailable(<logical> positions→,<logical> velocities→) Return true if positions and/or velocities are available.

tfamily logical scaleRadiiAvailable() Return true if scale radii are available.

tfamily logical spin3DAvailable() Return true if spin (vectors) are available.

tfamily logical spinAvailable() Return true if spin (magnitudes) are available.

tfamily void subhaloTrace(<double precision> time←,<double precision> position←,<double precision> velocity←) Supplies epochs, positions, and velocities for traced subhalos.

tfamily integer(kind=c_size_t) subhaloTraceCount() Returns the length of a node’s subhalo trace.

tfamily integer(kind=c_size_t) treeCount() Returns a count of the number of trees available.

tfamily integer(kind=kind_int8) treeIndex(<integer> i→) Returns the index of the $i^{th}$ tree.

tfamily logical treesAreSelfContained() Returns a Boolean integer specifying whether trees are self-contained.

tfamily integer treesHaveSubhalos() Returns a Boolean integer specifying whether or not the trees have subhalos.

tfamily double precision treeWeight(<integer> i→) Returns the weight to assign to the $i^{th}$ tree.

tfamily integer velocitiesIncludeHubbleFlow() Returns a Boolean integer specifying whether velocities include the Hubble flow.

tfamily logical velocityDispersionAvailable() Return true if halo velocity dispersions are available.

tfamily logical velocityMaximumAvailable() Return true if rotation curve velocity maxima are available.
16.5. Objects

tfamily mergerTreeImporterSussing

tfamily logical angularMomenta3DAvailable() Return true if angular momenta (vectors) are available.

tfamily logical angularMomentaAvailable() Return true if angular momenta (magnitudes) are available.

tfamily logical angularMomentaIncludeSubhalos() Returns a Boolean specifying whether halo angular momenta (or spins) include the contribution from their subhalos.

tfamily void close() Closes the file.

tfamily double precision cubeLength(<double precision> time→,<integer> status←) Returns the length of the simulation cube.

tfamily void import(<integer>i→,<logical> requireScaleRadii→,<logical> requireAngularMomenta→,<logical> requireAngularMomenta3D→,<logical> requireSpin→,<logical> requireSpin3D→,<logical> requirePositions→,<logical> requireParticleCounts→,<logical> requireVelocityMaxima→,<logical> requireVelocityDispersions→) Imports the $i^{th}$ tree.

tfamily logical zero inSubvolume(<double> x→, <double> y→, <double> z→, <logical> [buffered]→) Return true if the given $x,y,z$ position lies within the current subvolume (plus the buffer region if buffered is true.

tfamily logical zero inSubvolume1D(<double> x→, <integer> iSubvolume→, <logical> [buffered]→) Return true if the given $x$ position lies within the $iSubvolume$ subvolume (plus the buffer region if buffered is true.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily logical massesIncludeSubhalos() Returns a Boolean specifying whether halo masses include the contribution from their subhalos.

tfamily integer(kind=c_size_t) nodeCount(<integer>i→) Returns the number of nodes in the $i^{th}$ tree.

tfamily void open() Opens the file.

tfamily logical particleCountAvailable() Return true if particle counts are available.

tfamily integer positionsArePeriodic() Returns a Boolean integer specifying whether positions are periodic.

tfamily logical positionsAvailable(<logical> positions→,<logical> velocities→) Return true if positions and/or velocities are available.

tfamily logical scaleRadiiAvailable() Return true if scale radii are available.

tfamily logical spin3DAvailable() Return true if spin (vectors) are available.

tfamily logical spinAvailable() Return true if spin (magnitudes) are available.

tfamily void subhaloTrace(<double precision> time←,<double precision> position←,<double precision> velocity←) Supplies epochs, positions, and velocities for traced subhalos.

tfamily integer(kind=c_size_t) subhaloTraceCount() Returns the length of a node’s subhalo trace.

tfamily integer(kind=kind_int8) treeCount() Returns a count of the number of trees available.

tfamily integer(kind=kind_int8) treeIndex(<integer>i→) Returns the index of the $i^{th}$ tree.
tfamily integer treesAreSelfContained() Returns a Boolean integer specifying whether trees are self-contained.

tfamily integer treesHaveSubhalos() Returns a Boolean integer specifying whether or not the trees have subhalos.

tfamily double precision treeWeight(<integer> i →) Returns the weight to assign to the $i^{th}$ tree.

tfamily \logicalzero valueIsBad(<double> x →) Return true if the given x value is bad.

tfamily integer velocitiesIncludeHubbleFlow() Returns a Boolean integer specifying whether velocities include the Hubble flow.

tfamily logical velocityDispersionAvailable() Return true if halo velocity dispersions are available.

tfamily logical velocityMaximumAvailable() Return true if rotation curve velocity maxima are available.

tfamily nodeComponent
tfamily \doublezero density(<double(3)> positionSpherical→, <componentType> [componentType]→,
<massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<double> radius→, <componentType> [componentType]→,
<massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.
16.5. Objects

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

tfamily \void outputCount(<integer>integerPropertyCount ↔, <integer>doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily \void outputNames(<integer>integerProperty ↔, <char[*](:)>integerPropertyNames ↔, <char[*](:)>integerPropertyComments ↔, <double(:)>integerPropertyUnitsSI ↔, <integer>doubleProperty ↔, <char[*](:)>doublePropertyNames ↔, <char[*](:)>doublePropertyComments ↔, <double(:)>doublePropertyUnitsSI ↔, <double> time →, <integer>instance →) Generate names of outputtable properties.

tfamily \double zero potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

tfamily \double zero rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

tfamily \double zero rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

tfamily \int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

tfamily \double zero surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the surface density.

tfamily \textcolor{red}{\textless type(varying\textunderscore string)\textgreater} type() Return the type of this object.

tfamily nodeComponentAgeStatistics

tfamily \void builder(<*type(node)>componentDefinition →) Build a nodeComponent from a supplied XML definition.

tfamily \double zero density(<double(3)> positionSpherical →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the density.

tfamily \void deserializeRates(<double(:)> array ←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array ←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array ←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.
16. Coding GALACTICUS

tfamily \textcolor{red}{\textless double \textgreater} \text{diskIntegratedSFR()} Get the diskIntegratedSFR property of the ageStatistics component.

tfamily \textcolor{red}{\textless double \textgreater} \text{diskTimeWeightedIntegratedSFR()} Get the diskTimeWeightedIntegratedSFR property of the ageStatistics component.

tfamily \textcolor{red}{\textless double \textgreater} \text{diskTimeWeightedIntegratedSFRAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow)} Return a list of implementations that provide the given list of attributes for the diskTimeWeightedIntegratedSFR property of the ageStatistics component

tfamily \intzero \text{diskIntegratedSFRCount()} Compute the count of evolvable quantities in the diskIntegratedSFR property of the AgeStatisticsStandard component.

tfamily \textcolor{red}{\textless double \textgreater} \text{diskIntegratedSFRIsGettable()} Get the diskIntegratedSFR property of the ageStatistics component.

tfamily \logicalzero \text{diskIntegratedSFRIsSettable()} Specify whether the diskIntegratedSFR property of the ageStatistics component is settable.

tfamily \void \text{diskIntegratedSFRRate(<double> value)} Cumulate to the rate of the diskIntegratedSFR property of the AgeStatisticsStandard component.

tfamily \void \text{diskIntegratedSFRScale(<double> value)} Set the scale of the diskIntegratedSFR property of the AgeStatisticsStandard component.

tfamily \void \text{diskIntegratedSFRSet(<double> value)} Set the diskIntegratedSFR property of the ageStatistics component.

tfamily \void \text{dump()} Generate an ASCII dump of all properties.

tfamily \void \text{dumpRaw(<integer> fileHandle \rightarrow)} Generate a binary dump of all properties.
tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<double> radius\to, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the ageStatistics component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty↔, <char[*](:)> integerPropertyNames↔, <char[*](:)> integerPropertyComments↔, <double(*)> integerPropertyUnitsSI↔, <integer> doubleProperty↔, <char[*](:)> doublePropertyNames↔, <char[*](:)> doublePropertyComments↔, <double(*)> doublePropertyUnitsSI↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \doublezero potential(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the gravitational potential.

tfamily \doublezero rotationCurve(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:,:)> array→) Serialize the evolvable rates to an array.
### 16. Coding GALACTICUS

- `serializeScales(<double(:) array→)`: Serialize the evolvable scales to an array.
- `serializeValues(<double(:) array→)`: Serialize the evolvable quantities to an array.
- `spheroidIntegratedSFR()`: Get the `spheroidIntegratedSFR` property of the `ageStatistics` component.
- `spheroidIntegratedSFRAttributeMatch(<logical>[requireGettable→, <logical>[requireSettable→, <logical>[requireEvolvable→])`: Return a list of implementations that provide the given list of attributes for the `spheroidIntegratedSFR` property of the `ageStatistics` component.
- `spheroidIntegratedSFRCount()`: Compute the count of evolvable quantities in the `spheroidIntegratedSFR` property of the `ageStatisticsStandard` component.
- `spheroidIntegratedSFRIsGettable()`: Get the `spheroidIntegratedSFR` property of the `ageStatistics` component.
- `spheroidIntegratedSFRIsSettable()`: Specify whether the `spheroidIntegratedSFR` property of the `ageStatistics` component is settable.
- `spheroidIntegratedSFRRate(<double> value)`: Cumulate to the rate of the `spheroidIntegratedSFR` property of the `AgeStatisticsStandard` component.
- `spheroidIntegratedSFRScale(<double> value)`: Set the scale of the `spheroidIntegratedSFR` property of the `AgeStatisticsStandard` component.
- `spheroidIntegratedSFRSet(<double> value)`: Set the `spheroidIntegratedSFR` property of the `ageStatistics` component.
- `spheroidTimeWeightedIntegratedSFR()`: Get the `spheroidTimeWeightedIntegratedSFR` property of the `ageStatistics` component.
- `spheroidTimeWeightedIntegratedSFRAttributeMatch(<logical>[requireGettable→, <logical>[requireSettable→, <logical>[requireEvolvable→])`: Return a list of implementations that provide the given list of attributes for the `spheroidTimeWeightedIntegratedSFR` property of the `ageStatistics` component.
- `spheroidTimeWeightedIntegratedSFRCount()`: Compute the count of evolvable quantities in the `spheroidTimeWeightedIntegratedSFR` property of the `AgeStatisticsStandard` component.
- `spheroidTimeWeightedIntegratedSFRIsGettable()`: Get the `spheroidTimeWeightedIntegratedSFR` property of the `ageStatistics` component.
- `spheroidTimeWeightedIntegratedSFRIsSettable()`: Specify whether the `spheroidTimeWeightedIntegratedSFR` property of the `ageStatistics` component is settable.
- `spheroidTimeWeightedIntegratedSFRRate(<double> value)`: Cumulate to the rate of the `spheroidTimeWeightedIntegratedSFR` property of the `AgeStatisticsStandard` component.
- `spheroidTimeWeightedIntegratedSFRScale(<double> value)`: Set the scale of the `spheroidTimeWeightedIntegratedSFR` property of the `AgeStatisticsStandard` component.
- `spheroidTimeWeightedIntegratedSFRSet(<double> value)`: Set the `spheroidTimeWeightedIntegratedSFR` property of the `ageStatistics` component.
Objects

- **logical zero** `stdActive()` Return whether the standard implementation of the `ageStatistics` component class is active.

- **double zero** `surfDensity(<double(3) positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)` Compute the surface density.

- **text color red** `type()` Return the type of this object.

- `nodeComponentAgeStatisticsNull`
  - **void builder(<*type(node)> componentDefinition→)` Build a `nodeComponent` from a supplied XML definition.

- **double zero** `density(<double(3) positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)` Compute the density.

- **void deserializeRates(<double(:) array←)` Deserialize the evolvable rates from an array.

- **void deserializeScales(<double(:) array←)` Deserialize the evolvable scales from an array.

- **void deserializeValues(<double(:) array←)` Deserialize the evolvable quantities from an array.

- **void destroy()` Destroy the object.

- **text color red** `diskIntegratedSFR()` Get the `diskIntegratedSFR` property of the `ageStatistics` component.

- **text color red** `diskIntegratedSFRAttributeMatch(<logical> [requireGettable]→, <logical> [requireSettable]→, <logical> [requireEvolvable]→)` Return a list of implementations that provide the given list off attributes for the `diskIntegratedSFR` property of the `ageStatistics` component.

- **int zero** `diskIntegratedSFRCount()` Compute the count of evolvable quantities in the `diskIntegratedSFR` property of the `AgeStatisticsStandard` component.

- **text color red** `diskIntegratedSFRIsGettable()` Get the `diskIntegratedSFR` property of the `ageStatistics` component.

- **logical zero** `diskIntegratedSFRIsSettable()` Specify whether the `diskIntegratedSFR` property of the `ageStatistics` component is settable.

- **void diskIntegratedSFRRate(<double> value)` Cumulate to the rate of the `diskIntegratedSFR` property of the `AgeStatisticsStandard` component.

- **void diskIntegratedSFRScale(<double> value)` Set the scale of the `diskIntegratedSFR` property of the `AgeStatisticsStandard` component.

- **void diskIntegratedSFRSet(<double> value)` Set the `diskIntegratedSFR` property of the `ageStatistics` component.

- **text color red** `diskTimeWeightedIntegratedSFR()` Get the `diskTimeWeightedIntegratedSFR` property of the `ageStatistics` component.
16. Coding GALACTICUS

tfamily \textcolor{red}{\textless type(varying\_string):}\textgreater diskTimeWeightedIntegratedSFRAttributeMatch(<logical>[requireGettable]\rightarrow, <logical>[requireSettable]\rightarrow, <logical>[requireEvolvable]\rightarrow)

Return a list of implementations that provide the given list of attributes for the diskTimeWeightedIntegratedSFR property of the ageStatistics component.

tfamily \intzero diskTimeWeightedIntegratedSFRCount() Compute the count of evolvable quantities in the diskTimeWeightedIntegratedSFR property of the AgeStatisticsStandard component.

tfamily \textcolor{red}{\textless double\textgreater} diskTimeWeightedIntegratedSFRIsGettable() Get the diskTimeWeightedIntegratedSFR property of the ageStatistics component.

tfamily \logicalzero diskTimeWeightedIntegratedSFRIsSettable() Specify whether the diskTimeWeightedIntegratedSFR property of the ageStatistics component is settable.

tfamily \void diskTimeWeightedIntegratedSFRRate(<double> value) Cumulate to the rate of the diskTimeWeightedIntegratedSFR property of the AgeStatisticsStandard component.

tfamily \void diskTimeWeightedIntegratedSFRScale(<double> value) Set the scale of the diskTimeWeightedIntegratedSFR property of the AgeStatisticsStandard component.

tfamily \void diskTimeWeightedIntegratedSFRRate(<double> value) Set the diskTimeWeightedIntegratedSFR property of the ageStatistics component.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle\rightarrow) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<double> radius\rightarrow, <componentType>[componentType]\rightarrow, <massType>[massType]\rightarrow, <weightBy>[weightBy]\rightarrow, <integer>[weightIndex]\rightarrow, <logical>[haloLoaded]\rightarrow)

Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count\rightarrow, <varying_string>name\leftarrow) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the ageStatistics component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.
16.5. Objects

\begin{verbatim}
tfamily \void output\(<\text{integer}\>\text{integerProperty}\leftrightarrow, <\text{integer}\>\text{integerBufferCount}\leftrightarrow, <\text{integer}(\::,:)>\text{integerBuffer}\leftrightarrow, <\text{integer}\>\text{doubleProperty}\leftrightarrow, <\text{integer}\>\text{doubleBufferCount}\leftrightarrow, <\text{double}(\::,:)>\text{doubleBuffer}\leftrightarrow, <\text{double}\>\text{time}\rightarrow, <\text{integer}\>\text{instance}\rightarrow) Generate values of outputtable properties.

tfamily \void outputCount\(<\text{integer}\>\text{integerPropertyCount}\leftrightarrow, <\text{integer}\>\text{doublePropertyCount}\leftrightarrow, <\text{double}\>\text{time}\rightarrow, <\text{integer}\>\text{instance}\rightarrow) Compute a count of outputtable properties.

tfamily \void outputNames\(<\text{integer}\>\text{integerProperty}\leftrightarrow, <\text{char}\[*\](:)>\text{integerPropertyNames}\leftrightarrow, <\text{char}\[*\](:)>\text{integerPropertyComments}\leftrightarrow, <\text{double}(\::,:)>\text{integerPropertyUnitsSI}\leftrightarrow, <\text{integer}\>\text{doubleProperty}\leftrightarrow, <\text{char}\[*\](:)>\text{doublePropertyNames}\leftrightarrow, <\text{char}\[*\](:)>\text{doublePropertyComments}\leftrightarrow, <\text{double}(\::,:)>\text{doublePropertyUnitsSI}\leftrightarrow, <\text{double}\>\text{time}\rightarrow, <\text{integer}\>\text{instance}\rightarrow) Generate names of outputtable properties.

tfamily \doublezero potential\(<\text{double}\>\text{radius}\rightarrow, <\text{componentType}\>\text{componentType}\rightarrow, <\text{massType}\>\text{massType}\rightarrow, <\text{logical}\>\text{haloLoaded}\rightarrow) Compute the gravitational potential.

tfamily \void readRaw\(<\text{integer}\>\text{fileHandle}\rightarrow) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezero rotationCurve\(<\text{double}\>\text{radius}\rightarrow, <\text{componentType}\>\text{componentType}\rightarrow, <\text{massType}\>\text{massType}\rightarrow, <\text{logical}\>\text{haloLoaded}\rightarrow) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient\(<\text{double}\>\text{radius}\rightarrow, <\text{componentType}\>\text{componentType}\rightarrow, <\text{massType}\>\text{massType}\rightarrow, <\text{logical}\>\text{haloLoaded}\rightarrow) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates\(<\text{double}(\::,:)>\text{array}\rightarrow) Serialize the evolvable rates to an array.

tfamily \void serializeScales\(<\text{double}(\::,:)>\text{array}\rightarrow) Serialize the evolvable scales to an array.

tfamily \void serializeValues\(<\text{double}(\::,:)>\text{array}\rightarrow) Serialize the evolvable quantities to an array.

\textcolor{red}{<\text{double}> spheroidIntegratedSFR()} Get the spheroidIntegratedSFR property of the ageStatistics component.

\textcolor{red}{<\text{variable>(varying\_type)(::)}> spheroidIntegratedSFRAttributeMatch\(<\text{logical}\>\text{requireGettable}\rightarrow, <\text{logical}\>\text{requireSettable}\rightarrow, <\text{logical}\>\text{requireEvolvable}\rightarrow) Return a list of implementations that provide the given list of attributes for the spheroidIntegratedSFR property of the ageStatistics component.

\intzero spheroidIntegratedSFRCount() Compute the count of evolvable quantities in the spheroidIntegratedSFR property of the AgeStatisticsStandard component.

\textcolor{red}{<\text{variable}> spheroidIntegratedSFRIsGettable()} Get the spheroidIntegratedSFR property of the ageStatistics component.

\logicalzero spheroidIntegratedSFRIsSettable() Specify whether the spheroidIntegratedSFR property of the ageStatistics component is settable.

\void spheroidIntegratedSFRRate\(<\text{double}\>\text{value}) Cumulate to the rate of the spheroidIntegratedSFR property of the AgeStatisticsStandard component.

\void spheroidIntegratedSFRScale\(<\text{double}\>\text{value}) Set the scale of the spheroidIntegratedSFR property of the AgeStatisticsStandard component.
\end{verbatim}
16. Coding GALACTICUS

tfamily \void spheroidIntegratedSFRSet(<double> value) Set the \texttt{spheroidIntegratedSFR} property of the \texttt{ageStatistics} component.

tfamily \textcolor{red}{\textless\textgreater} spheroidTimeWeightedIntegratedSFR() Get the \texttt{spheroidTimeWeightedIntegratedSFR} property of the \texttt{ageStatistics} component.

tfamily \textcolor{red}{\textless\textgreater} spheroidTimeWeightedIntegratedSFRAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list off attributes for the \texttt{spheroidTimeWeightedIntegratedSFR} property of the \texttt{ageStatistics} component.

tfamily \intzero spheroidTimeWeightedIntegratedSFRCount() Compute the count of evolvable quantities in the \texttt{spheroidTimeWeightedIntegratedSFR} property of the \texttt{AgeStatisticsStandard} component.

tfamily \textcolor{red}{\textless\textgreater} spheroidTimeWeightedIntegratedSFRIsGettable() Get the \texttt{spheroidTimeWeightedIntegratedSFR} property of the \texttt{ageStatistics} component.

tfamily \logicalzero spheroidTimeWeightedIntegratedSFRIsSettable() Specify whether the \texttt{spheroidTimeWeightedIntegratedSFR} property of the \texttt{ageStatistics} component is settable.

tfamily \void spheroidTimeWeightedIntegratedSFRRate(<double> value) Cumulate to the rate of the \texttt{spheroidTimeWeightedIntegratedSFR} property of the \texttt{AgeStatisticsStandard} component.

tfamily \void spheroidTimeWeightedIntegratedSFRScale(<double> value) Set the scale of the \texttt{spheroidTimeWeightedIntegratedSFR} property of the \texttt{AgeStatisticsStandard} component.

tfamily \void spheroidTimeWeightedIntegratedSFRSet(<double> value) Set the \texttt{spheroidTimeWeightedIntegratedSFR} property of the \texttt{ageStatistics} component.

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the \texttt{ageStatistics} component class is active.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical \rightarrow, <componentType> \rightarrow, <massType> \rightarrow, <weightBy> \rightarrow, <integer> \rightarrow, <logical> \rightarrow) Compute the surface density.

tfamily \textcolor{red}{\textless\textgreater} type() Return the type of this object.

tfamily nodeComponentAgeStatisticsStandard

tfamily \void builder(<\texttt{type(node)}\textgreater componentDefinition \rightarrow) Build a \texttt{nodeComponent} from a supplied XML definition.

tfamily \doublezero density(<double(3)> positionSpherical \rightarrow, <componentType> \rightarrow, <massType> \rightarrow, <weightBy> \rightarrow, <integer> \rightarrow, <logical> \rightarrow) Compute the density.

tfamily \void deserializeRates(<double(:)> array \leftarrow) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array \leftarrow) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array \leftarrow) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.
16.5. Objects

tfamily \textcolor{red}{\textless double \textgreater} diskIntegratedSFR() Get the diskIntegratedSFR property of the ageStatistics component.

tfamily \textcolor{red}{\textless type(varying\_string):(\textgreater)} diskIntegratedSFRAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the diskIntegratedSFR property of the ageStatistics component.

tfamily \intzerodiskIntegratedSFRCount() Compute the count of evolvable quantities in the diskIntegratedSFR property of the AgeStatisticsStandard component.

tfamily \textcolor{red}{\textless double \textgreater} diskIntegratedSFRIsGettable() Get the diskIntegratedSFR property of the ageStatistics component.

tfamily \logicalzerodiskIntegratedSFRIsSettable() Specify whether the diskIntegratedSFR property of the ageStatistics component is settable.

tfamily \void diskIntegratedSFRRate(<double> value) Cumulate to the rate of the diskIntegratedSFR property of the AgeStatisticsStandard component.

tfamily \void diskIntegratedSFRScale(<double> value) Set the scale of the diskIntegratedSFR property of the AgeStatisticsStandard component.

tfamily \void diskIntegratedSFRRSet(<double> value) Set the diskIntegratedSFR property of the ageStatistics component.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.
tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(\double radius\rightarrow, \componentType[componentType]\rightarrow, \massType[massType]\rightarrow, \weightBy[weightBy]\rightarrow, \integer[weightIndex]\rightarrow, \logical[haloLoaded]\rightarrow) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host TreeNode object.

tfamily \void hotHaloCoolingAbundancesRate(\type(abundances) value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(\double value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(\double value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(\integer count\rightarrow, \varying_string name\leftarrow) Return the name of a property given is index.

tfamily \logical zero nullIsActive() Return whether the null implementation of the ageStatistics component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(\integer integerProperty\leftrightarrow, \integer integerBufferCount\leftrightarrow, \integer[\ldots]\leftrightarrow integerBuffer\leftrightarrow, \integer doubleProperty\leftrightarrow, \integer doubleBufferCount\leftrightarrow, \double[\ldots]\leftrightarrow doubleBuffer\leftrightarrow, \double time\rightarrow, \integer instance\rightarrow) Generate values of outputtable properties.

tfamily \void outputCount(\integer integerPropertyCount\leftrightarrow, \integer doublePropertyCount\leftrightarrow, \double time\rightarrow, \integer instance\rightarrow) Compute a count of outputtable properties.

tfamily \void outputNames(\integer integerProperty\leftrightarrow, \char[\ldots] integerPropertyNames\leftrightarrow, \char[\ldots] integerPropertyComments\leftrightarrow, \double integerPropertyUnitsSI\leftrightarrow, \integer doubleProperty\leftrightarrow, \char[\ldots] doublePropertyNames\leftrightarrow, \char[\ldots] doublePropertyComments\leftrightarrow, \double[\ldots] doublePropertyUnitsSI\leftrightarrow, \double time\rightarrow, \integer instance\rightarrow) Generate names of outputtable properties.

tfamily \double zero potential(\double radius\rightarrow, \componentType[componentType]\rightarrow, \massType[massType]\rightarrow, \logical[haloLoaded]\rightarrow) Compute the gravitational potential.

tfamily \void readRaw(\integer fileHandle\rightarrow) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezero rotationCurve(\double radius\rightarrow, \componentType[componentType]\rightarrow, \massType[massType]\rightarrow, \logical[haloLoaded]\rightarrow) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(\double radius\rightarrow, \componentType[componentType]\rightarrow, \massType[massType]\rightarrow, \logical[haloLoaded]\rightarrow) Compute the rotation curve gradient.

tfamily \int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.
16.5. Objects

tfamily \void serializeRates(<double()> array) Serialize the evolvable rates to an array.
tfamily \void serializeScales(<double()> array) Serialize the evolvable scales to an array.
tfamily \void serializeValues(<double()> array) Serialize the evolvable quantities to an array.
tfamily {\textcolor{red}{\textless double\textgreater}spheroidIntegratedSFR()} Get the spheroidIntegratedSFR property of the ageStatistics component.
tfamily {\textcolor{red}{\textless double\textgreater}spheroidIntegratedSFRAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →)} Return a list of implementations that provide the given list of attributes for the spheroidIntegratedSFR property of the ageStatistics component.
tfamily \intzero spheroidIntegratedSFRCount() Compute the count of evolvable quantities in the spheroidIntegratedSFR property of the AgeStatisticsStandard component.
tfamily {\textcolor{red}{\textless double\textgreater}spheroidIntegratedSFRIsGettable()} Get the spheroidIntegratedSFR property of the ageStatistics component.
tfamily \logicalzerospheroidIntegratedSFRIsSettable() Specify whether the spheroidIntegratedSFR property of the ageStatistics component is settable.
tfamily \void spheroidIntegratedSFRRate(<double> value) Cumulate to the rate of the spheroidIntegratedSFR property of the AgeStatisticsStandard component.
tfamily \void spheroidIntegratedSFRScale(<double> value) Set the scale of the spheroidIntegratedSFR property of the AgeStatisticsStandard component.
tfamily \void spheroidIntegratedSFRSet(<double> value) Set the spheroidIntegratedSFR property of the ageStatistics component.
tfamily {\textcolor{red}{\textless double\textgreater}spheroidTimeWeightedIntegratedSFR()} Get the spheroidTimeWeightedIntegratedSFR property of the ageStatistics component.
tfamily {\textcolor{red}{\textless type(varying\_string)\textgreater}spheroidTimeWeightedIntegratedSFRAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →)} Return a list of implementations that provide the given list of attributes for the spheroidTimeWeightedIntegratedSFR property of the ageStatistics component.
tfamily \intzero spheroidTimeWeightedIntegratedSFRCount() Compute the count of evolvable quantities in the spheroidTimeWeightedIntegratedSFR property of the AgeStatisticsStandard component.
tfamily {\textcolor{red}{\textless double\textgreater}spheroidTimeWeightedIntegratedSFRIsGettable()} Get the spheroidTimeWeightedIntegratedSFR property of the ageStatistics component.
tfamily \logicalzerospheroidTimeWeightedIntegratedSFRIsSettable() Specify whether the spheroidTimeWeightedIntegratedSFR property of the ageStatistics component is settable.
tfamily \void spheroidTimeWeightedIntegratedSFRRate(<double> value) Cumulate to the rate of the spheroidTimeWeightedIntegratedSFR property of the AgeStatisticsStandard component.
tfamily \void spheroidTimeWeightedIntegratedSFRScale(<double> value) Set the scale of the spheroidTimeWeightedIntegratedSFR property of the AgeStatisticsStandard component.
tfamily \ void spheroidTimeWeightedIntegratedSFRSet(\texttt{<double> value}) Set the spheroidTimeWeightedIntegratedSFR property of the \texttt{ageStatistics} component.

tfamily \ logicalzero standardIsActive() Return whether the standard implementation of the ageStatistics component class is active.

tfamily \ doublezero surfaceDensity(\texttt{<double(3)> positionCylindrical\rightarrow, <componentType>} \ [componentType]\rightarrow, \texttt{<massType>} \ [massType]\rightarrow, \texttt{<weightBy>} \ [weightBy]\rightarrow, \texttt{<integer>} \ [weightIndex]\rightarrow, \texttt{<logical>} \ [haloLoaded]\rightarrow) Compute the surface density.

tfamily \ textcolor{red}\textless type(varying\_\_string)\textgreater\ type() Return the type of this object.

tfamily nodeComponentBasic

tfamily \ textcolor{red}\textless double\textgreater\ accretionRate() Get the accretionRate property of the basic component.

tfamily \ textcolor{red}\textless type(varying\_\_string)\textgreater;\texttt{<logical>}\texttt{requireGettable}\rightarrow, \texttt{<logical>}\texttt{requireSettable}\rightarrow, \texttt{<logical>}\texttt{requireEvolvable}\rightarrow) Return a list of implementations that provide the given list off attributes for the accretionRate property of the basic component

tfamily \ textcolor{red}\textless double\textgreater\ accretionRateIsGettable() Get the accretionRate property of the basic component.

tfamily \ logicalzero accretionRateIsSettable() Specify whether the accretionRate property of the basic component is settable.

tfamily \ void accretionRateSet(\texttt{<double> value}) Set the accretionRate property of the basic component.

tfamily \ void builder(\texttt{<\text{\texttt{type(node)}}>componentDefinition\rightarrow}) Build a nodeComponent from a supplied XML definition.

tfamily \ doublezero density(\texttt{<double(3)> positionSpherical\rightarrow, <componentType>} \ [componentType]\rightarrow, \texttt{<massType>} \ [massType]\rightarrow, \texttt{<weightBy>} \ [weightBy]\rightarrow, \texttt{<integer>} \ [weightIndex]\rightarrow, \texttt{<logical>} \ [haloLoaded]\rightarrow) Compute the density.

tfamily \ void deserializeRates(\texttt{<double(:)> array\leftarrow}) Deserialize the evolvable rates from an array.

tfamily \ void deserializeScales(\texttt{<double(:)> array\leftarrow}) Deserialize the evolvable scales from an array.

tfamily \ void deserializeValues(\texttt{<double(:)> array\leftarrow}) Deserialize the evolvable quantities from an array.

tfamily \ void destroy() Destroy the object.

tfamily \ void dump() Generate an ASCII dump of all properties.

tfamily \ void dumpRaw(\texttt{<integer> fileHandle\rightarrow}) Generate a binary dump of all properties.

tfamily \ void dumpXML() Generate an XML dump of all properties.

tfamily \ doublezero enclosedMass(\texttt{<double> radius\rightarrow, <componentType>} \ [componentType]\rightarrow, \texttt{<massType>} \ [massType]\rightarrow, \texttt{<weightBy>} \ [weightBy]\rightarrow, \texttt{<integer>} \ [weightIndex]\rightarrow, \texttt{<logical>} \ [haloLoaded]\rightarrow) Compute the mass enclosed within a radius.
tfamily \textcolor{red}{	extless *type(treeNode)\textgreater} host() Return a pointer to the host
treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the
tHotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the
tHotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}{\textless double\textgreater} mass() Get the mass property of the basic component.

tfamily \void massAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the mass property of the basic component

tfamily \intzero massCount() Compute the count of evolvable quantities in the mass property of the BasicStandard component.

tfamily \textcolor{red}{\textless double\textgreater} massIsGettable() Get the mass property of the basic component.

tfamily \logicalzero massIsSettable() Specify whether the mass property of the basic component is settable.

tfamily \textcolor{red}{\textless double\textgreater} massMaximum() Get the massMaximum property of the basic component.

tfamily \void massMaximumSet(<double> value) Set the massMaximum property of the basic component.

tfamily \logicalzero massMaximumIsSettable() Specify whether the massMaximum property of the basic component is settable.

tfamily \void massMaximumSet(<double> value) Set the massMaximum property of the basic component.

tfamily \void massRate(<double> value) Cumulate to the rate of the mass property of the BasicStandard component.

tfamily \void massScale(<double> value) Set the scale of the mass property of the BasicStandard component.

tfamily \void massSet(<double> value) Set the mass property of the basic component.

tfamily \void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.
tfamily \logicalzero nonEvolvingIsActive() Return whether the nonEvolving implementation of the basic component class is active.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the basic component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty, <integer> integerBufferCount, <integer(:,:)> integerBuffer, <integer> doubleProperty, <integer> doubleBufferCount, <double(:,:)> doubleBuffer, <double> time, <integer> instance) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount, <integer> doublePropertyCount, <double> time, <integer> instance) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty, <char[*](:)> integerPropertyNames, <char[*](:)> integerPropertyComments, <double(:)> integerPropertyUnitsSI, <integer> doubleProperty, <char[*](:)> doublePropertyNames, <char[*](:)> doublePropertyComments, <double(:)> doublePropertyUnitsSI, <double> time, <integer> instance) Generate names of outputtable properties.

tfamily \doublezero potential(<double> radius, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

tfamily \doublezero rotationCurve(<double> radius, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double> radius, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array) Serialize the evolvable quantities to an array.

16. Coding GALACTICUS

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the basic component class is active.

tfamily \logicalzero standardTrackingIsActive() Return whether the standardTracking implementation of the basic component class is active.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.

tfamily \textcolor{red}{\textless double\textgreater} time() Get the time property of the basic component.

394
tfamily \textcolor{red}{\textless double\textgreater} \texttt{timeCount()} Compute the count of evolvable quantities in the \texttt{time} property of the \texttt{BasicNonEvolving} component.

tfamily \textcolor{red}{\textless double\textgreater} \texttt{timeIsGettable()} Get the \texttt{time} property of the \texttt{basic} component.

tfamily \textcolor{red}{\textless double\textgreater} \texttt{timeIsSettable()} Specify whether the \texttt{time} property of the \texttt{basic} component is settable.

tfamily \textcolor{red}{\textless double\textgreater} \texttt{timeLastIsolated()} Get the \texttt{timeLastIsolated} property of the \texttt{basic} component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)} \texttt{timeLastIsolatedAttributeMatch(\textless logical\textgreater [requireGettable] \rightarrow, \textless logical\textgreater [requireSettable] \rightarrow, \textless logical\textgreater [requireEvolvable] \rightarrow)} Return a list of implementations that provide the given list of attributes for the \texttt{timeLastIsolated} property of the \texttt{basic} component.

tfamily \textcolor{red}{\textless double\textgreater} \texttt{timeLastIsolatedIsGettable()} Get the \texttt{timeLastIsolated} property of the \texttt{basic} component.

tfamily \textcolor{red}{\textless double\textgreater} \texttt{timeLastIsolatedIsSettable()} Specify whether the \texttt{timeLastIsolated} property of the \texttt{basic} component is settable.

tfamily \texttt{timeLastIsolatedSet(\textless double\textgreater \texttt{value})} Set the \texttt{timeLastIsolated} property of the \texttt{basic} component.

tfamily \texttt{timeRate(\textless double\textgreater \texttt{value})} Cumulate to the rate of the \texttt{time} property of the \texttt{BasicNonEvolving} component.

tfamily \texttt{timeScale(\textless double\textgreater \texttt{value})} Set the scale of the \texttt{time} property of the \texttt{BasicNonEvolving} component.

tfamily \texttt{timeSet(\textless double\textgreater \texttt{value})} Set the \texttt{time} property of the \texttt{basic} component.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} \texttt{type()} Return the type of this object.

tfamily nodeComponentBasicNonEvolving

tfamily \textcolor{red}{\textless double\textgreater} \texttt{accretionRate()} Get the \texttt{accretionRate} property of the \texttt{basic} component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)} \texttt{accretionRateAttributeMatch(\textless logical\textgreater [requireGettable] \rightarrow, \textless logical\textgreater [requireSettable] \rightarrow, \textless logical\textgreater [requireEvolvable] \rightarrow)} Return a list of implementations that provide the given list of attributes for the \texttt{accretionRate} property of the \texttt{basic} component.

tfamily \textcolor{red}{\textless double\textgreater} \texttt{accretionRateIsGettable()} Get the \texttt{accretionRate} property of the \texttt{basic} component.

tfamily \textcolor{red}{\textless double\textgreater} \texttt{accretionRateIsSettable()} Specify whether the \texttt{accretionRate} property of the \texttt{basic} component is settable.

tfamily \texttt{accretionRateSet(\textless double\textgreater \texttt{value})} Set the \texttt{accretionRate} property of the \texttt{basic} component.

tfamily \texttt{builder(\textless type(node)\textgreater componentDefinition \rightarrow)} Build a \texttt{nodeComponent} from a supplied XML definition.
tfamily \doublezero density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}{\textless double\textgreater} mass() Get the mass property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} massIsGettable() Get the mass property of the basic component.

tfamily \logicalzero massIsSettable() Specify whether the mass property of the basic component is settable.

tfamily \textcolor{red}{\textless double\textgreater} massCount() Compute the count of evolvable quantities in the mass property of the BasicStandard component.

tfamily \textcolor{red}{\textless double\textgreater} massMaximum() Get the massMaximum property of the basic component.
16.5. Objects

tfamily\[textcolor{red}\{\textless varying\_\ string\(\textgreater\}\}\textgreater massMaximumAttributeMatch(\textless logical\[requireGettable\]→, \textless logical\[requireSettable\]→, \textless logical\[requireEvolvable\]→) Return a list of implementations that provide the given list of attributes for the massMaximum property of the basic component.

tfamily\[textcolor{red}\{\textless double\textgreater\} massMaximumIsGettable() Get the massMaximum property of the basic component.

tfamily logicalzero massMaximumIsSettable() Specify whether the massMaximum property of the basic component is settable.

tfamily void massMaximumSet(\textless double\(\textgreater\) value) Set the massMaximum property of the basic component.

tfamily void massRate(\textless double\(\textgreater\) value) Cumulate to the rate of the mass property of the BasicStandard component.

tfamily void massScale(\textless double\(\textgreater\) value) Set the scale of the mass property of the BasicStandard component.

tfamily void massSet(\textless double\(\textgreater\) value) Set the mass property of the basic component.

tfamily void nameFromIndex(\textless integer\(\textgreater\) count→, \textless varying\_string\(\textgreater\) name←) Return the name of a property given its index.

tfamily logicalzero nonEvolvingIsActive() Return whether the nonEvolving implementation of the basic component class is active.

tfamily logicalzero nullIsActive() Return whether the null implementation of the basic component class is active.

tfamily void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily void output(\textless integer\(\textgreater\) integerProperty↔, \textless integer\(\textgreater\) integerBufferCount↔, \textless integer(\(\textgreater\)) integerBuffer↔, \textless integer\(\textgreater\) doubleProperty↔, \textless integer\(\textgreater\) doubleBufferCount↔, \textless double(\(\textgreater\)) doubleBuffer↔, \textless double\(\textgreater\) time→, \textless integer\(\textgreater\) instance→) Generate values of outputtable properties.

tfamily void outputCount(\textless integer\(\textgreater\) integerPropertyCount↔, \textless integer\(\textgreater\) doublePropertyCount↔, \textless double\(\textgreater\) time→, \textless integer\(\textgreater\) instance→) Compute a count of outputtable properties.

tfamily void outputNames(\textless integer\(\textgreater\) integerProperty↔, \textless char\[\ast\](\(\textgreater\)) integerPropertyNameNames↔, \textless char\[\ast\](\(\textgreater\)) integerPropertyComments↔, \textless double\(\textgreater\) integerPropertyUnitsSI↔, \textless integer\(\textgreater\) doubleProperty↔, \textless char\[\ast\](\(\textgreater\)) doublePropertyNameNames↔, \textless char\[\ast\](\(\textgreater\)) doublePropertyComments↔, \textless double\(\textgreater\) doublePropertyUnitsSI↔, \textless double\(\textgreater\) time→, \textless integer\(\textgreater\) instance→) Generate names of outputtable properties.

tfamily doublezero potential(\textless double\(\textgreater\) radius→, \textless componentType\(\textgreater\) [componentType]→, \textless massType\(\textgreater\) [massType]→, \textless logical\[haloLoaded\]→) Compute the gravitational potential.

tfamily void readRaw(\textless integer\(\textgreater\) fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily doublezero rotationCurve(\textless double\(\textgreater\) radius→, \textless componentType\(\textgreater\) [componentType]→, \textless massType\(\textgreater\) [massType]→, \textless logical\[haloLoaded\]→) Compute the rotation curve.
16. Coding GALACTICUS

tfamily double zero rotationCurveGradient(<double> radius\rightarrow, <componentType> [componentType]\rightarrow, <massType> [massType]\rightarrow, <logical> [haloLoaded]\rightarrow) Compute the rotation curve gradient.

tfamily\int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily\void serializeRates(<double(:)> array\rightarrow) Serialize the evolvable rates to an array.

tfamily\void serializeScales(<double(:)> array\rightarrow) Serialize the evolvable scales to an array.

tfamily\void serializeValues(<double(:)> array\rightarrow) Serialize the evolvable quantities to an array.

tfamily\logical zero standardIsActive() Return whether the standard implementation of the basic component class is active.

tfamily\logical zero standardTrackingIsActive() Return whether the standardTracking implementation of the basic component class is active.

tfamily double zero surfaceDensity(<double(3)> positionCylindrical\rightarrow, <componentType> [componentType]\rightarrow, <massType> [massType]\rightarrow, <weightBy> [weightBy]\rightarrow, <integer> [weightIndex]\rightarrow, <logical> [haloLoaded]\rightarrow) Compute the surface density.

tfamily \textcolor{red}{\textless double\textgreater} time() Get the time property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} timeAttributeMatch(<logical>[requireGettable]\rightarrow, <logical>[requireSettable]\rightarrow, <logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list off attributes for the time property of the basic component

tfamily\int zero timeCount() Compute the count of evolvable quantities in the time property of the BasicNonEvolving component.

tfamily \textcolor{red}{\textless double\textgreater} timeIsGettable() Get the time property of the basic component.

tfamily\logical zero timeIsSettable() Specify whether the time property of the basic component is settable.

tfamily \textcolor{red}{\textless double\textgreater} timeLastIsolated() Get the timeLastIsolated property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} timeLastIsolatedAttributeMatch(<logical>[requireGettable]\rightarrow, <logical>[requireSettable]\rightarrow, <logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list off attributes for the timeLastIsolated property of the basic component

tfamily \textcolor{red}{\textless double\textgreater} timeLastIsolatedIsGettable() Get the timeLastIsolated property of the basic component.

tfamily\logical zero timeLastIsolatedIsSettable() Specify whether the timeLastIsolated property of the basic component is settable.

tfamily\void timeLastIsolatedSet(<double> value) Set the timeLastIsolated property of the basic component.

tfamily\void timeRate(<double> value) Cumulate to the rate of the time property of the BasicNonEvolving component.
tfamily \void timeScale(<double> value) Set the scale of the time property of the BasicNonEvolving component.

tfamily \void timeSet(<double> value) Set the time property of the basic component.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

tfamily nodeComponentBasicNull

tfamily \textcolor{red}{\textless double\textgreater} accretionRate() Get the accretionRate property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} accretionRateAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the accretionRate property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} accretionRateIsGettable() Get the accretionRate property of the basic component.

tfamily \logicalzero accretionRateIsSettable() Specify whether the accretionRate property of the basic component is settable.

tfamily \void accretionRateSet(<double> value) Set the accretionRate property of the basic component.

tfamily \void builder(<*type(node)>componentDefinition) Build a nodeComponent from a supplied XML definition.

tfamily \doublezero density(<double(3)> positionSpherical→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.
tfamily \textcolor{red}{\textless double\textgreater} mass() Get the mass property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} massAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the mass property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} massIsGettable() Get the mass property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} massIsSettable() Specify whether the mass property of the basic component is settable.

tfamily \textcolor{red}{\textless double\textgreater} massMaximum() Get the massMaximum property of the basic component.

400
16.5. Objects

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:,)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

tfamily \void outputCount(<integer>integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily \void outputNames(<integer>integerProperty ↔, <char[*](:)>integerPropertyNames ↔, <char[*](:)>integerPropertyComments ↔, <double(:,:,)>integerPropertyUnitsSI ↔, <integer>doubleProperty ↔, <char[*](:)>doublePropertyNames ↔, <char[*](:)>doublePropertyComments ↔, <double(:,:,)>doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

tfamily \double zero potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

tfamily \void readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \double zero rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

tfamily \double zero rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

tfamily \logical zero standardIsActive() Return whether the standard implementation of the basic component class is active.

tfamily \logical zero standardTrackingIsActive() Return whether the standardTracking implementation of the basic component class is active.

tfamily \double zero surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the surface density.

tfamily \textcolor{red}{\textless double\textgreater} time() Get the time property of the basic component.

tfamily \textcolor{red}{\textless varying\_string(,:)\textgreater} timeAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the time property of the basic component.

tfamily \intzero timeCount() Compute the count of evolvable quantities in the time property of the BasicNonEvolving component.
16. Coding Galacticus

tfamily \textcolor{red}{\textless double\textgreater} timeIsGettable() Get the time property of the basic component.

tfamily \logicalzero timeIsSettable() Specify whether the time property of the basic component is settable.

tfamily \textcolor{red}{\textless double\textgreater} timeLastIsolated() Get the timeLastIsolated property of the basic component.

tfamily \textcolor{red}{\textlesstype(varying_string)(:)} timeLastIsolatedAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list off attributes for the timeLastIsolated property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} timeLastIsolatedIsGettable() Get the timeLastIsolated property of the basic component.

tfamily \logicalzero timeLastIsolatedIsSettable() Specify whether the timeLastIsolated property of the basic component is settable.

tfamily \void timeLastIsolatedSet(<double> value) Set the timeLastIsolated property of the basic component.

tfamily \void timeRate(<double> value) Cumulate to the rate of the time property of the BasicNonEvolving component.

tfamily \void timeScale(<double> value) Set the scale of the time property of the BasicNonEvolving component.

tfamily \void timeSet(<double> value) Set the time property of the basic component.

tfamily \textcolor{red}{\textless type(varying_string)\textgreater} type() Return the type of this object.

tfamily nodeComponentBasicStandard

tfamily \textcolor{red}{\textless double\textgreater} accretionRate() Get the accretionRate property of the basic component.

tfamily \textcolor{red}{\textlesstype(varying_string)(:)} accretionRateAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list off attributes for the accretionRate property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} accretionRateIsGettable() Get the accretionRate property of the basic component.

tfamily \logicalzero accretionRateIsSettable() Specify whether the accretionRate property of the basic component is settable.

tfamily \void accretionRateSet(<double> value) Set the accretionRate property of the basic component.

tfamily \void builder(<\*type(node)>componentDefinition \rightarrow) Build a nodeComponent from a supplied XML definition.
16.5. Objects

tfamily \textcolor{red}{\textless double\textgreater} \textcolor{red}{\textless double\textgreater} density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

tfamily \textcolor{red}{\textless double\textgreater} deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \textcolor{red}{\textless double\textgreater} deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \textcolor{red}{\textless double\textgreater} deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \textcolor{red}{\textless double\textgreater} destroy() Destroy the object.

tfamily \textcolor{red}{\textless double\textgreater} deserializeRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \textcolor{red}{\textless double\textgreater} dumpXML() Generate an XML dump of all properties.

tfamily \textcolor{red}{\textless double\textgreater} enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless double\textgreater} mass() Get the mass property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} massAttributeMatch(<logical> [requireGettable]→, <logical> [requireSettable]→, <logical> [requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the mass property of the basic component

tfamily \textcolor{red}{\textless double\textgreater} massCount() Compute the count of evolvable quantities in the mass property of the BasicStandard component.


**Coding GALACTICUS**

```plaintext
tfamily \textcolor{red}{\textless \textless} varying\_string(:)\textgreater \ massMaximumAttributeMatch\left(<\text{logical}>[\text{requireGettable}\rightarrow, <\text{logical}>[\text{requireSettable}\rightarrow, <\text{logical}>[\text{requireEvolvable}\rightarrow)\text{ Return a list of implementations that provide the given list of attributes for the massMaximum property of the basic component.}

tfamily \textcolor{red}{\textless \textless} double\textgreater \ massMaximumIsGettable() Get the massMaximum property of the basic component.

tfamily \textcolor{red}{\textless \textless} double\textgreater \ massMaximumIsSettable() Specify whether the massMaximum property of the basic component is settable.

tfamily \textless \textless \textgreater \ massMaximumSet\left(<\text{double}> value\right) Set the massMaximum property of the basic component.

tfamily \textless \textless \textgreater \ massRate\left(<\text{double}> value\right) Cumulate to the rate of the mass property of the BasicStandard component.

tfamily \textless \textless \textgreater \ massScale\left(<\text{double}> value\right) Set the scale of the mass property of the BasicStandard component.

tfamily \textless \textless \textgreater \ massSet\left(<\text{double}> value\right) Set the mass property of the basic component.

tfamily \textcolor{red}{\textless \textless} integer\textgreater \ nameFromIndex\left(<\text{integer}> count\rightarrow, <\text{varying\_string}> name\leftarrow\right) Return the name of a property given its index.

tfamily \textcolor{red}{\textless \textless} logical\textgreater \ nonEvolvingIsActive() Return whether the nonEvolving implementation of the basic component class is active.

tfamily \textcolor{red}{\textless \textless} logical\textgreater \ nullIsActive() Return whether the null implementation of the basic component class is active.

tfamily \textless \textless \textgreater \ odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \textless \textless \textgreater \ odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \textless \textless \textgreater \ output\left(<\text{integer}> integerProperty\leftrightarrow, <\text{integer}> integerBufferCount\leftrightarrow, <\text{integer}(:,:)> integerBuffer\leftrightarrow, <\text{integer}> doubleProperty\leftrightarrow, <\text{integer}> doubleBufferCount\leftrightarrow, <\text{double}(:,:)> doubleBuffer\leftrightarrow, <\text{double}> time\rightarrow, <\text{integer}> instance\rightarrow\right) Generate values of outputtable properties.

tfamily \textless \textless \textgreater \ outputCount\left(<\text{integer}> integerPropertyCount\leftrightarrow, <\text{integer}> doublePropertyCount\leftrightarrow, <\text{double}> time\rightarrow, <\text{integer}> instance\rightarrow\right) Compute a count of outputtable properties.

tfamily \textless \textless \textgreater \ outputNames\left(<\text{integer}> integerProperty\leftrightarrow, <\text{char}\[*\]> integerPropertyNames\leftrightarrow, <\text{char}\[*\]> integerPropertyComments\leftrightarrow, <\text{double}(:,:)> integerPropertyUnitsSI\leftrightarrow, <\text{integer}> doubleProperty\leftrightarrow, <\text{char}\[*\]> doublePropertyNames\leftrightarrow, <\text{char}\[*\]> doublePropertyComments\leftrightarrow, <\text{double}(:,:)> doublePropertyUnitsSI\leftrightarrow, <\text{double}> time\rightarrow, <\text{integer}> instance\rightarrow\right) Generate names of outputtable properties.

tfamily \textless \textless \textgreater \ potential\left(<\text{double}> radius\rightarrow, <\text{componentType}> [\text{componentType}\rightarrow, <\text{massType}> [\text{massType}\rightarrow, <\text{logical}> [\text{haloLoaded}\rightarrow]\right) Compute the gravitational potential.

tfamily \textless \textless \textgreater \ readRaw\left(<\text{integer}> fileHandle\rightarrow\right) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \textless \textless \textgreater \ rotationCurve\left(<\text{double}> radius\rightarrow, <\text{componentType}> [\text{componentType}\rightarrow, <\text{massType}> [\text{massType}\rightarrow, <\text{logical}> [\text{haloLoaded}\rightarrow]\right) Compute the rotation curve.
```
16.5. Objects

\texttt{tfamily doublezero rotationCurveGradient(\textless double\textgreater \; radius\rightarrow, \textless componentType\textgreater \; \texttt{[componentType]}\rightarrow,} \\
\texttt{\textless massType\textgreater \; \texttt{[massType]}\rightarrow, \texttt{logical}\rightarrow \texttt{[haloLoaded]}\rightarrow) Compute the rotation curve gradient.}

\texttt{tfamily intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.}

\texttt{tfamily void serializeRates(\textless double(:)\textgreater \; array\rightarrow) Serialize the evolvable rates to an array.}

\texttt{tfamily void serializeScales(\textless double(:)\textgreater \; array\rightarrow) Serialize the evolvable scales to an array.}

\texttt{tfamily void serializeValues(\textless double(:)\textgreater \; array\rightarrow) Serialize the evolvable quantities to an array.}

\texttt{tfamily logicalzero standardIsActive() Return whether the standard implementation of the basic component class is active.}

\texttt{tfamily logicalzero standardTrackingIsActive() Return whether the standardTracking implementation of the basic component class is active.}

\texttt{tfamily doublezero surfaceDensity(\textless double(3)\textgreater \; positionCylindrical\rightarrow, \textless componentType\textgreater \; \texttt{[componentType]}\rightarrow,} \\
\texttt{\textless massType\textgreater \; \texttt{[massType]}\rightarrow, \texttt{weightBy}\rightarrow \texttt{[weightBy]}\rightarrow, \texttt{integer}\rightarrow \texttt{[weightIndex]}\rightarrow,} \\
\texttt{\texttt{logical}\rightarrow \texttt{[haloLoaded]}\rightarrow) Compute the surface density.}

\texttt{tfamily \textcolor{red}{\textlesstype(varying\_string)(:)}\textgreater timeAttributeMatch(\texttt{logical}\rightarrow \texttt{[requireGettable]}\rightarrow,} \\
\texttt{\texttt{logical}\rightarrow \texttt{[requireSettable]}\rightarrow, \texttt{logical}\rightarrow \texttt{[requireEvolvable]}\rightarrow) Return a list of implementations that provide the given list off attributes for the \texttt{time} property of the \texttt{basic} component.}

\texttt{tfamily \intzero timeCount() Compute the count of evolvable quantities in the \texttt{time} property of the \texttt{BasicNonEvolving} component.}

\texttt{tfamily \textcolor{red}{\textlesstype(double)}\textgreater timeIsGettable() Get the \texttt{time} property of the \texttt{basic} component.}

\texttt{tfamily \textcolor{red}{\textlesstype(double)} timeLastIsolated() Get the \texttt{timeLastIsolated} property of the \texttt{basic} component.}

\texttt{tfamily \textcolor{red}{\textlesstype(double)}\textgreater timeLastIsolatedIsGettable() Get the \texttt{timeLastIsolated} property of the \texttt{basic} component.}

\texttt{tfamily \textcolor{red}{\textlesstype(double)}\textgreater timeLastIsolatedSet(<double> value) Set the \texttt{timeLastIsolated} property of the \texttt{basic} component.}

\texttt{tfamily \textcolor{red}{\textlesstype(double)}\textgreater timeRate(<double> value) Cumulate to the rate of the \texttt{time} property of the \texttt{BasicNonEvolving} component.}
16. Coding Galacticus

tfamily \void timeScale(<double> value) Set the scale of the time property of the BasicNonEvolving component.

tfamily \void timeSet(<double> value) Set the time property of the basic component.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

ntfamily nodeComponentBasicStandardTracking

tfamily \textcolor{red}{\textless double\textgreater} accretionRate() Get the accretionRate property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} accretionRateAttributeMatch(<logical>[requireGettable]$→$, <logical>[requireSettable]$→$, <logical>[requireEvolvable]$→$) Return a list of implementations that provide the given list off attributes for the accretionRate property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} accretionRateIsGettable() Get the accretionRate property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} accretionRateIsSettable() Specify whether the accretionRate property of the basic component is settable.

tfamily \void accretionRateSet(<double> value) Set the accretionRate property of the basic component.

tfamily \void builder(<*type(node)>componentDefinition$→$) Build a nodeComponent from a supplied XML definition.

tfamily \double zero density(<double(3)> positionSpherical$→$, <componentType>[componentType]$→$, <massType>[massType]$→$, <weightBy>[weightBy]$→$, <integer>[weightIndex]$→$, <logical>[haloLoaded]$→$) Compute the density.

tfamily \void deserializeRates(<double(:)> array$←$) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array$←$) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array$←$) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle$→$) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \double zero enclosedMass(<double>radius$→$, <componentType>[componentType]$→$, <massType>[massType]$→$, <weightBy>[weightBy]$→$, <integer>[weightIndex]$→$, <logical>[haloLoaded]$→$) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.
16.5. Objects

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}{\textless double\textgreater} mass() Get the mass property of the basic component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)} massAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the mass property of the basic component.

tfamily \int massCount() Compute the count of evolvable quantities in the mass property of the BasicStandard component.

tfamily \textcolor{red}{\textless double\textgreater} massIsGettable() Get the mass property of the basic component.

tfamily \logical massIsSettable() Specify whether the mass property of the basic component is settable.

tfamily \textcolor{red}{\textless double\textgreater} massMaximum() Get the massMaximum property of the basic component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)} massMaximumAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the massMaximum property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} massMaximumIsGettable() Get the massMaximum property of the basic component.

tfamily \logical massMaximumIsSettable() Specify whether the massMaximum property of the basic component is settable.

tfamily \void massMaximumSet(<double> value) Set the massMaximum property of the basic component.

tfamily \void massRate(<double> value) Cumulate to the rate of the mass property of the BasicStandard component.

tfamily \void massScale(<double> value) Set the scale of the mass property of the BasicStandard component.

tfamily \void massSet(<double> value) Set the mass property of the basic component.

tfamily \void nameFromIndex(<integer> count→, <varying\_string> name←) Return the name of a property given is index.

tfamily \logical nonEvolvingIsActive() Return whether the nonEvolving implementation of the basic component class is active.

tfamily \logical nullIsActive() Return whether the null implementation of the basic component class is active.
tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty, <integer> integerBufferCount, <integer> integerBuffer, <integer> doubleProperty, <integer> doubleBufferCount, <double> time, <integer> instance) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount, <integer> doublePropertyCount, <double> time, <integer> instance) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty, <char(*)[:] integerPropertyNames, <char(*)[:] integerPropertyComments, <double(*)[:] integerPropertyUnitsSI, <integer> doubleProperty, <char(*)[:] doublePropertyNames, <char(*)[:] doublePropertyComments, <double(*)[:] doublePropertyUnitsSI, <double> time, <integer> instance) Generate names of outputtable properties.

tfamily \doublezeropotential(<double> radius, <componentType> [componentType], <massType> [massType], <logical> [haloLoaded]) Compute the gravitational potential.

tfamily \void readRaw(<integer> fileHandle) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezerorotationCurve(<double> radius, <componentType> [componentType], <massType> [massType], <logical> [haloLoaded]) Compute the rotation curve.

tfamily \doublezerorotationCurveGradient(<double> radius, <componentType> [componentType], <massType> [massType], <logical> [haloLoaded]) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double> array) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double> array) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double> array) Serialize the evolvable quantities to an array.

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the basic component class is active.

tfamily \logicalzero standardTrackingIsActive() Return whether the standardTracking implementation of the basic component class is active.

tfamily \doublezerosurfaceDensity(<double(3)> positionCylindrical, <componentType> [componentType], <massType> [massType], <weightBy> [weightBy], <integer> [weightIndex], <logical> [haloLoaded]) Compute the surface density.

tfamily \textcolor{red}{\textless double\textgreater} time() Get the time property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} timeAttributeMatch(<logical> [requireGettable], <logical> [requireSettable], <logical> [requireEvolvable]) Return a list of implementations that provide the given list off attributes for the time property of the basic component.

tfamily \intzero timeCount() Compute the count of evolvable quantities in the time property of the BasicNonEvolving component.
16.5. Objects

tfamily \textcolor{red}{\textless double\textgreater} timeIsGettable() Get the time property of the basic component.

tfamily \logicalzero timeIsSettable() Specify whether the time property of the basic component is settable.

tfamily \textcolor{red}{\textless double\textgreater} timeLastIsolated() Get the timeLastIsolated property of the basic component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater timeLastIsolatedAttributeMatch(<logical>[requireGettable] → , <logical>[requireSettable] → , <logical>[requireEvolvable] → ) Return a list of implementations that provide the given list of attributes for the timeLastIsolated property of the basic component.

tfamily \textcolor{red}{\textless double\textgreater} timeLastIsolatedIsGettable() Get the timeLastIsolated property of the basic component.

tfamily \logicalzero timeLastIsolatedIsSettable() Specify whether the timeLastIsolated property of the basic component is settable.

tfamily \void timeLastIsolatedSet(<double> value) Set the timeLastIsolated property of the basic component.

tfamily \void timeRate(<double> value) Cumulate to the rate of the time property of the BasicNonEvolving component.

tfamily \void timeScale(<double> value) Set the scale of the time property of the BasicNonEvolving component.

tfamily \void timeSet(<double> value) Set the time property of the basic component.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

{\family nodeComponentBlackHole

tfamily \textcolor{red}{\textless double\textgreater} accretionRate() Get the accretionRate property of the blackHole component.

tfamily \textcolor{red}{\textless type(varying\_string)::\textgreater} accretionRateAttributeMatch(<logical>[requireGettable] → , <logical>[requireSettable] → , <logical>[requireEvolvable] → ) Return a list of implementations that provide the given list of attributes for the accretionRate property of the blackHole component.

tfamily \textcolor{red}{\textless double\textgreater} accretionRateIsGettable() Get the accretionRate property of the blackHole component.

tfamily \logicalzero accretionRateIsSettable() Specify whether the accretionRate property of the blackHole component is settable.

tfamily \void builder(<\&type(node)>componentDefinition → ) Build a nodeComponent from a supplied XML definition.

tfamily \doublezero density(<double(3)> positionSpherical → , <componentType> [componentType] → , <massType> [massType] → , <weightBy> [weightBy] → , <integer> [weightIndex] → , <logical> [haloLoaded] → ) Compute the density.

tfamily \void deserializeRates(<double(:)> array ← ) Deserialize the evolvable rates from an array.
tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \double zero enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}{\textless double\textgreater} mass() Get the mass property of the blackHole component.

tfamily \textcolor{red}{\textless type(varying\_string):\textgreater} massAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the mass property of the blackHole component.

tfamily \int zero massCount() Compute the count of evolvable quantities in the mass property of the BlackHoleSimple component.

tfamily \textcolor{red}{\textless double\textgreater} massIsGettable() Get the mass property of the blackHole component.

tfamily \logical zero massIsSettable() Specify whether the mass property of the blackHole component is settable.

tfamily \void massRate(<double> value) Cumulate to the rate of the mass property of the BlackHoleSimple component.

tfamily \void massScale(<double> value) Set the scale of the mass property of the BlackHoleSimple component.

tfamily \textcolor{red}{\textless double\textgreater} massSeed() Get the massSeed property of the blackHole component.
16.5. Objects

tfamily \textcolor{red}{\textless textless type(varying\_string)():\textgreater} massSeedAttributeMatch(<logical>[requireGettable]→←<logical>[requireSettable]→←<logical>[requireEvolvable]→←) Return a list of implementations that provide the given list off attributes for the massSeed property of the blackHole component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:):\textgreater} massSeedGettable() Get the massSeed property of the blackHole component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:):\textgreater} massSeedIsGettable() Get the massSeed property of the blackHole component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:):\textgreater} massSeedIsSettable() Specify whether the massSeed property of the blackHole component is settable.


tfamily \void massSet(<double> value) Set the mass property of the blackHole component.

tfamily \void nameFromIndex(<integer> count→←<varying_string=name←) Return the name of a property given is index.


tfamily \logical zero nullIsActive() Return whether the null implementation of the blackHole component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.


tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.


tfamily \void output(<integer> integerProperty←←<integer> integerBufferCount←←<integer(:,:)> integerBuffer←←<integer> doubleProperty←←<integer> doubleBufferCount←←<double(:,:)> doubleBuffer←<double> time→←<integer> instance→←) Generate values of outputtable properties.


tfamily \void outputCount(<integer> integerPropertyCount←←<integer> doublePropertyCount←←<double> time→←<integer> instance→←) Compute a count of outputtable properties.


tfamily \void outputNames(<integer> integerProperty←←<char[*](:)> integerPropertyNames←←<char[*](:)> integerPropertyComments←←<double(:)> integerPropertyUnitsSI←←<integer> doubleProperty←←<char[*](:)> doublePropertyNames←←<char[*](:)> doublePropertyComments←←<double(:)> doublePropertyUnitsSI←<double> time→←<integer> instance→←) Generate names of outputtable properties.


tfamily \int zero radialPositionCount() Compute the count of evolvable quantities in the radialPosition property of the BlackHoleStandard component.


tfamily \textcolor{red}{\textlesstype(varying\_string)(:):\textgreater} radialPositionAttributeMatch(<logical>[requireGettable]→←<logical>[requireSettable]→←<logical>[requireEvolvable]→←) Return a list of implementations that provide the given list off attributes for the radialPosition property of the blackHole component.


tfamily \int zero radialPositionIsGettable() Compute the count of evolvable quantities in the radialPosition property of the BlackHoleStandard component.


tfamily \textcolor{red}{\textlesstype(varying\_string)(:):\textgreater} radialPositionIsGettable() Get the radialPosition property of the blackHole component.


tfamily \logical zero radialPositionIsSettable() Specify whether the radialPosition property of the blackHole component is settable.


tfamily \void radialPositionRate(<double> value) Cumulate to the rate of the radialPosition property of the BlackHoleStandard component.
16. Coding Galacticus

tfamily \void \textcolor{red}{\textless\textbf{double}\textgreater} \text{radialPositionScale)(<\textbf{double}> value) Set the scale of the \textbf{radialPosition} property of the BlackHoleStandard component.

tfamily \void \text{radialPositionSet(<\textbf{double}> value) Set the \textbf{radialPosition} property of the blackHole component.}

tfamily \textcolor{red}{\textless\textbf{double}\textgreater} \text{radiativeEfficiency() Get the \textbf{radiativeEfficiency} property of the blackHole component.}

tfamily \textcolor{red}{\textless\textbf{type(varying\_string):}\textgreater} \text{radiativeEfficiencyAttributeMatch(<\textbf{logical}>[requireGettable]\rightarrow, <\textbf{logical}>[requireSettable]\rightarrow, <\textbf{logical}>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the \textbf{radiativeEfficiency} property of the blackHole component.}

tfamily \textcolor{red}{\textless\textbf{double}\textgreater} \text{radiativeEfficiencyIsGettable() Get the \textbf{radiativeEfficiency} property of the blackHole component.}

tfamily \textbf{logical zero} \text{radiativeEfficiencyIsSettable() Specify whether the \textbf{radiativeEfficiency} property of the blackHole component is settable.}

tfamily \textbf{double zero} \text{rotationCurve(<\textbf{double}> radius\rightarrow, <\textbf{componentType}>[\textbf{componentType}]\rightarrow, <\textbf{massType}>[\textbf{massType}]\rightarrow, <\textbf{logical}>[\textbf{haloLoaded}]\rightarrow) Compute the rotation curve.}

tfamily \textbf{double zero} \text{rotationCurveGradient(<\textbf{double}> radius\rightarrow, <\textbf{componentType}>[\textbf{componentType}]\rightarrow, <\textbf{massType}>[\textbf{massType}]\rightarrow, <\textbf{logical}>[\textbf{haloLoaded}]\rightarrow) Compute the rotation curve gradient.}

tfamily \intzero \text{serializeCount() Return a count of the number of evolvable quantities to be evolved.}

tfamily \void \text{serializeRates(<\textbf{double}(:)> array\rightarrow) Serialize the evolvable rates to an array.}

tfamily \void \text{serializeScales(<\textbf{double}(:)> array\rightarrow) Serialize the evolvable scales to an array.}

tfamily \void \text{serializeValues(<\textbf{double}(:)> array\rightarrow) Serialize the evolvable quantities to an array.}

tfamily \textbf{logical zero} \text{simpleIsActive() Return whether the simple implementation of the blackHole component class is active.}

tfamily \textcolor{red}{\textless\textbf{double}\textgreater} \text{spin() Get the \textbf{spin} property of the blackHole component.}

tfamily \textcolor{red}{\textless\textbf{type(varying\_string):}\textgreater} \text{spinAttributeMatch(<\textbf{logical}>[requireGettable]\rightarrow, <\textbf{logical}>[requireSettable]\rightarrow, <\textbf{logical}>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the \textbf{spin} property of the blackHole component.}

tfamily \intzero \text{spinCount() Compute the count of evolvable quantities in the \textbf{spin} property of the BlackHoleStandard component.}

tfamily \textcolor{red}{\textless\textbf{double}\textgreater} \text{spinIsGettable() Get the \textbf{spin} property of the blackHole component.}

tfamily \textbf{logical zero} \text{spinIsSettable() Specify whether the \textbf{spin} property of the blackHole component is settable.}

tfamily \void \text{spinRate(<\textbf{double}> value) Cumulate to the rate of the \textbf{spin} property of the BlackHoleStandard component.}
16.5. Objects

tfamily \textcolor{red}{\textless double\textgreater} spinScale() Set the scale of the spin property of the BlackHoleStandard component.

tfamily \textcolor{red}{\textless double\textgreater} spinSeed() Get the spinSeed property of the blackHole component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} spinSeedAttributeMatch(<logical>[requireGettable], <logical>[requireSettable], <logical>[requireEvolvable]) Return a list of implementations that provide the given list of attributes for the spinSeed property of the blackHole component.

tfamily \textcolor{red}{\textless double\textgreater} spinSeedIsGettable() Get the spinSeed property of the blackHole component.

tfamily \textcolor{red}{\textless double\textgreater} spinSeedIsSettable() Specify whether the spinSeed property of the blackHole component is settable.

tfamily \textcolor{red}{\textless double\textgreater} spinSet() Set the spin property of the blackHole component.

standardIsActive() Return whether the standard implementation of the blackHole component class is active.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} type() Return the type of this object.

nodeComponentBlackHoleNull

tfamily \textcolor{red}{\textless double\textgreater} accretionRate() Get the accretionRate property of the blackHole component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} accretionRateAttributeMatch(<logical>[requireGettable], <logical>[requireSettable], <logical>[requireEvolvable]) Return a list of implementations that provide the given list of attributes for the accretionRate property of the blackHole component.

tfamily \textcolor{red}{\textless double\textgreater} accretionRateIsGettable() Get the accretionRate property of the blackHole component.
tfamily \logicalzero accretionRateIsSettable() Specify whether the accretionRate property of the blackHole component is settable.

tfamily \void builder(<\text{type(node)}>componentDefinition\rightarrow) Build a nodeComponent from a supplied XML definition.

tfamily \doublezero density(<\double(3)> positionSpherical\rightarrow, <\text{componentType}> [componentType]\rightarrow, <\text{massType}> [massType]\rightarrow, <\text{weightBy}> [weightBy]\rightarrow, <\integer> [weightIndex]\rightarrow, <\logical> [haloLoaded]\rightarrow) Compute the density.

tfamily \void deserializeRates(<\double(:)> array\leftarrow) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<\double(:)> array\leftarrow) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<\double(:)> array\leftarrow) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<\integer> fileHandle\rightarrow) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<\double> radius\rightarrow, <\text{componentType}> [componentType]\rightarrow, <\text{massType}> [massType]\rightarrow, <\text{weightBy}> [weightBy]\rightarrow, <\integer> [weightIndex]\rightarrow, <\logical> [haloLoaded]\rightarrow) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless \text{type(treeNode)}\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<\text{type(abundances)}> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<\double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<\double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}{\textless \text{double}\textgreater} mass() Get the mass property of the blackHole component.

tfamily \textcolor{red}{\textless \text{double}\textgreater\textless \text{varying\_string}()\textgreater} massAttributeMatch(<\logical> [requireGettable]\rightarrow, <\logical> [requireSettable]\rightarrow, <\logical> [requireEvolvable]\rightarrow) Return a list of implementations that provide the given list off attributes for the mass property of the blackHole component

tfamily \intzero massCount() Compute the count of evolvable quantities in the mass property of the BlackHoleSimple component.

tfamily \textcolor{red}{\textless \text{double}\textgreater} massIsGettable() Get the mass property of the blackHole component.

tfamily \logicalzero massIsSettable() Specify whether the mass property of the blackHole component is settable.
16.5. Objects

tfamily \ void massRate(<double> value) Cumulate to the rate of the mass property of the BlackHoleSimple component.

tfamily \ void massScale(<double> value) Set the scale of the mass property of the BlackHoleSimple component.

tfamily \ textcolor{red}\textless double\textgreater massSeed() Get the massSeed property of the blackHole component.

tfamily \ textcolor{red}\textless double\textgreater massSeedAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the massSeed property of the blackHole component

tfamily \ textcolor{red}\textless double\textgreater massSeedIsGettable() Get the massSeed property of the blackHole component.

tfamily \ logical zero massSeedIsSettable() Specify whether the massSeed property of the blackHole component is settable.

tfamily \ void massSet(<double> value) Set the mass property of the blackHole component.

tfamily \ void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

tfamily \ logical zero nullIsActive() Return whether the null implementation of the blackHole component class is active.

tfamily \ void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \ void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \ void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)>integerBuffer↔, <integer>doubleProperty↔, <integer>doubleBufferCount↔, <double(:,:)>doubleBuffer↔, <double> time→, <integer>instance→) Generate values of outputtable properties.

tfamily \ void outputCount(<integer>integerPropertyCount↔, <integer>doublePropertyCount↔, <double> time→, <integer>instance→) Compute a count of outputtable properties.

tfamily \ void outputNames(<integer>integerProperty↔, <char\[*\](:)>integerPropertyNames↔, <char\[*\](:)>integerPropertyComments↔, <double(:)>integerPropertyUnitsSI↔, <integer>doubleProperty↔, <char\[*\](:)>doublePropertyNames↔, <char\[*\](:)>doublePropertyComments↔, <double(:)>doublePropertyUnitsSI↔, <double> time→, <integer>instance→) Generate names of outputtable properties.

tfamily \ double zero potential(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the gravitational potential.

tfamily \ textcolor{red}\textless double\textgreater radialPosition() Get the radialPosition property of the blackHole component.

tfamily \ textcolor{red}\textless double\textgreater radialPositionAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the radialPosition property of the blackHole component

tfamily \ int zero radialPositionCount() Compute the count of evolvable quantities in the radialPosition property of the BlackHoleStandard component.
tfamily \textcolor{red}{\textless double\textgreater} radialPositionIsGettable() Get the radialPosition property of the blackHole component.

tfamily \textcolor{red}{\textless double\textgreater} radialPositionIsSettable() Specify whether the radialPosition property of the blackHole component is settable.

tfamily \textcolor{red}{\textless double\textgreater} radialPositionRate(<double> value) Cumulate to the rate of the radialPosition property of the BlackHoleStandard component.

tfamily \textcolor{red}{\textless double\textgreater} radialPositionScale(<double> value) Set the scale of the radialPosition property of the BlackHoleStandard component.

tfamily \textcolor{red}{\textless double\textgreater} radialPositionSet(<double> value) Set the radialPosition property of the blackHole component.

tfamily \textcolor{red}{\textless double\textgreater} radiativeEfficiency() Get the radiativeEfficiency property of the blackHole component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater radiativeEfficiencyAttributeMatch(<logical> [requireGettable] →, <logical> [requireSettable] →, <logical> [requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the radiativeEfficiency property of the blackHole component.

tfamily \textcolor{red}{\textless double\textgreater} radiativeEfficiencyIsGettable() Get the radiativeEfficiency property of the blackHole component.

tfamily \textcolor{red}{\textless double\textgreater} radiativeEfficiencyIsSettable() Specify whether the radiativeEfficiency property of the blackHole component is settable.

tfamily \textcolor{red}{\textless double\textgreater} spin() Get the spin property of the blackHole component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater spinAttributeMatch(<logical> [requireGettable] →, <logical> [requireSettable] →, <logical> [requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the spin property of the blackHole component.
16.5. Objects

tfamily \intzero spinCount() Compute the count of evolvable quantities in the spin property of the BlackHoleStandard component.

tfamily \textcolor{red}{\textless\textgreater\textcolor{red}{\textless\textgreater}\textcolor{red}{\textless\textgreater}} spinIsGettable() Get the spin property of the blackHole component.

tfamily \logicalzero spinIsSettable() Specify whether the spin property of the blackHole component is settable.

tfamily \void spinRate(<double> value) Cumulate to the rate of the spin property of the BlackHoleStandard component.

tfamily \void spinScale(<double> value) Set the scale of the spin property of the BlackHoleStandard component.

tfamily \textcolor{red}{\textless\textgreater\textcolor{red}{\textless\textgreater}} spinSeed() Get the spinSeed property of the blackHole component.

tfamily \textcolor{red}{\textless\textgreater\textcolor{red}{\textless\textgreater}} spinSeedAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the spinSeed property of the blackHole component.

tfamily \void spinSeedIsGettable() Get the spinSeed property of the blackHole component.

tfamily \logicalzero spinSeedIsSettable() Specify whether the spinSeed property of the blackHole component is settable.

tfamily \void spinSet(<double> value) Set the spin property of the blackHole component.

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the blackHole component class is active.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the surface density.

tfamily \textcolor{red}{\textless\textgreater\textcolor{red}{\textless\textgreater}} tripleInteractionTime() Get the tripleInteractionTime property of the blackHole component.

tfamily \textcolor{red}{\textless\textgreater\textcolor{red}{\textless\textgreater}} tripleInteractionTimeAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the tripleInteractionTime property of the blackHole component.

tfamily \textcolor{red}{\textless\textgreater\textcolor{red}{\textless\textgreater}} tripleInteractionTimeIsGettable() Get the tripleInteractionTime property of the blackHole component.

tfamily \logicalzero tripleInteractionTimeIsSettable() Specify whether the tripleInteractionTime property of the blackHole component is settable.

tfamily \void tripleInteractionTimeSet(<double> value) Set the tripleInteractionTime property of the blackHole component.

tfamily \textcolor{red}{\textless\textgreater\textcolor{red}{\textless\textgreater}} type() Return the type of this object.
16. Coding GALACTICUS

tfamily nodeComponentBlackHoleSimple
tfamily \textcolor{red}{\textless \textgreater \textgreater} accretionRate() Get the \texttt{accretionRate} property of the \texttt{blackHole} component.

tfamily \textcolor{red}{\textless \textgreater \textgreater} accretionRateAttributeMatch(\texttt{<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→}) Return a list of implementations that provide the given list off attributes for the \texttt{accretionRate} property of the \texttt{blackHole} component.

tfamily \textcolor{red}{\textless \textgreater \textgreater} accretionRateIsGettable() Get the \texttt{accretionRate} property of the \texttt{blackHole} component.

tfamily \textcolor{red}{\textless \textgreater \textgreater} accretionRateIsSettable() Specify whether the \texttt{accretionRate} property of the \texttt{blackHole} component is settable.

tfamily \texttt{builder(\texttt{<type(node)>componentDefinition→}) Build a nodeComponent from a supplied XML definition.

tfamily \texttt{density(\texttt{<double(3)> positionSpherical→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→}) Compute the density.

tfamily \texttt{deserializeRates(\texttt{<double(:)> array←}) Deserialize the evolvable rates from an array.

tfamily \texttt{deserializeScales(\texttt{<double(:)> array←}) Deserialize the evolvable scales from an array.

tfamily \texttt{deserializeValues(\texttt{<double(:)> array←}) Deserialize the evolvable quantities from an array.

tfamily \texttt{destroy()} Destroy the object.

tfamily \texttt{dump()} Generate an ASCII dump of all properties.

tfamily \texttt{dumpRaw(\texttt{<integer> fileHandle→}) Generate a binary dump of all properties.

tfamily \texttt{dumpXML()} Generate an XML dump of all properties.

tfamily \texttt{enclosedMass(\texttt{<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→}) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless \textgreater \textgreater} host() Return a pointer to the host treeNode object.

tfamily \texttt{hotHaloCoolingAbundancesRate(\texttt{<type(abundances)> value}) Cumulate to the rate of the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component.

tfamily \texttt{hotHaloCoolingAngularMomentumRate(\texttt{<double> value}) Cumulate to the rate of the \texttt{hotHaloCoolingAngularMomentum} property of the \texttt{hotHalo} component.

tfamily \texttt{hotHaloCoolingMassRate(\texttt{<double> value}) Cumulate to the rate of the \texttt{hotHaloCoolingMass} property of the \texttt{hotHalo} component.

tfamily \texttt{initialize()} Initialize the object.

tfamily \textcolor{red}{\textless \textgreater \textgreater} mass() Get the \texttt{mass} property of the \texttt{blackHole} component.
tfamily \textcolor{red}{\textless\textgreater} massAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the mass property of the blackHole component.

tfamily \intzero massCount() Compute the count of evolvable quantities in the mass property of the BlackHoleSimple component.

tfamily \textcolor{red}{\textless\textgreater} massIsGettable() Get the mass property of the blackHole component.

tfamily \logicalzero massIsSettable() Specify whether the mass property of the blackHole component is settable.

tfamily \void massRate(<double> value) Cumulate to the rate of the mass property of the BlackHoleSimple component.

tfamily \void massScale(<double> value) Set the scale of the mass property of the BlackHoleSimple component.

tfamily \textcolor{red}{\textless\textgreater} massSeed() Get the massSeed property of the blackHole component.

tfamily \textcolor{red}{\textless\textgreater} massSeedAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the massSeed property of the blackHole component.

tfamily \textcolor{red}{\textless\textgreater} massSeedIsGettable() Get the massSeed property of the blackHole component.

tfamily \logicalzero massSeedIsSettable() Specify whether the massSeed property of the blackHole component is settable.

tfamily \void massSet(<double> value) Set the mass property of the blackHole component.

tfamily \void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given an index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the blackHole component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(;)>) integerBuffer↔, <integer> doubleProperty↔, <integer> doubleBufferCount↔, <double(;)>) doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty↔, <char(*)>] integerPropertyNames↔, <char(*)]) integerPropertyComments↔, <double(;)>) integerPropertyUnitsSI↔, <integer> doubleProperty↔, <char(*)>] doublePropertyNames↔, <char(*)]) doublePropertyComments↔, <double(;)>) doublePropertyUnitsSI↔, <double> time→, <integer> instance→) Generate names of outputtable properties.
tfamily double zero potential(<double> radius \rightarrow, <componentType> |componentType| \rightarrow, <massType> |massType| \rightarrow, <logical> |haloLoaded| \rightarrow) Compute the gravitational potential.

tfamily \textcolor{red}{\textless double\textgreater} radialPosition() Get the radialPosition property of the BlackHole component.

tfamily \textcolor{red}{\textless double\textgreater} radialPositionAttributeMatch(<logical>|requireGettable| \rightarrow, <logical>|requireSettable| \rightarrow, <logical>|requireEvolvable| \rightarrow) Return a list of implementations that provide the given list of attributes for the radialPosition property of the BlackHole component.

tfamily \intzero radialPositionCount() Compute the count of evolvable quantities in the radialPosition property of the BlackHoleStandard component.

tfamily \textcolor{red}{\textless double\textgreater} radialPositionIsGettable() Get the radialPosition property of the BlackHole component.

tfamily logical zero radialPositionIsSettable() Specify whether the radialPosition property of the BlackHole component is settable.

tfamily \void radialPositionRate(<double> value) Cumulate to the rate of the radialPosition property of the BlackHoleStandard component.

tfamily \void radialPositionScale(<double> value) Set the scale of the radialPosition property of the BlackHole component.

tfamily \void radialPositionSet(<double> value) Set the radialPosition property of the BlackHole component.

tfamily \textcolor{red}{\textless double\textgreater} radiativeEfficiency() Get the radiativeEfficiency property of the BlackHole component.

tfamily \textcolor{red}{\textless type(varying\_string)(;)\textgreater} radiativeEfficiencyAttributeMatch(<logical>|requireGettable| \rightarrow, <logical>|requireSettable| \rightarrow, <logical>|requireEvolvable| \rightarrow) Return a list of implementations that provide the given list of attributes for the radiativeEfficiency property of the BlackHole component.

tfamily \textcolor{red}{\textless double\textgreater} radiativeEfficiencyIsGettable() Get the radiativeEfficiency property of the BlackHole component.

tfamily logical zero radiativeEfficiencyIsSettable() Specify whether the radiativeEfficiency property of the BlackHole component is settable.

tfamily \void readRaw(<integer> fileHandle \rightarrow) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily double zero rotationCurve(<double> radius \rightarrow, <componentType> |componentType| \rightarrow, <massType> |massType| \rightarrow, <logical> |haloLoaded| \rightarrow) Compute the rotation curve.

tfamily \void rotationCurveGradient(<double> radius \rightarrow, <componentType> |componentType| \rightarrow, <massType> |massType| \rightarrow, <logical> |haloLoaded| \rightarrow) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array \rightarrow) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array \rightarrow) Serialize the evolvable scales to an array.
tfamily \void serializeValues(<double(:)> array \rightarrow) Serialize the evolvable quantities to an array.

tfamily \logicalzero simpleIsActive() Return whether the simple implementation of the blackHole component class is active.

tfamily \textcolor{red}{\textless double\textgreater} spin() Get the spin property of the blackHole component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:\textgreater)} spinAttributeMatch(<logical> [requireGettable] \rightarrow, <logical> [requireSettable] \rightarrow, <logical> [requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the spin property of the blackHole component.

tfamily \intzero spinCount() Compute the count of evolvable quantities in the spin property of the BlackHoleStandard component.

tfamily \textcolor{red}{\textless double\textgreater} spinIsGettable() Get the spin property of the blackHole component.

tfamily \logicalzero spinIsSettable() Specify whether the spin property of the blackHole component is settable.

tfamily \void spinRate(<double> value) Cumulate to the rate of the spin property of the BlackHoleStandard component.

tfamily \void spinScale(<double> value) Set the scale of the spin property of the BlackHoleStandard component.

tfamily \textcolor{red}{\textless double\textgreater} spinSeed() Get the spinSeed property of the blackHole component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:\textgreater)} spinSeedAttributeMatch(<logical> [requireGettable] \rightarrow, <logical> [requireSettable] \rightarrow, <logical> [requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the spinSeed property of the blackHole component.

tfamily \void spinSet(<double> value) Set the spin property of the blackHole component.

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the blackHole component class is active.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <weightBy> [weightBy] \rightarrow, <integer> [weightIndex] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the surface density.

tfamily \textcolor{red}{\textless double\textgreater} tripleInteractionTime() Get the tripleInteractionTime property of the blackHole component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:\textgreater)} tripleInteractionTimeAttributeMatch(<logical> [requireGettable] \rightarrow, <logical> [requireSettable] \rightarrow, <logical> [requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the tripleInteractionTime property of the blackHole component.
16. Coding GALACTICUS

tfamily \textcolor{red}{\textless double\textgreater} tripleInteractionTimeIsGettable() Get the \texttt{tripleInteractionTime} property of the \texttt{blackHole} component.

tfamily \textcolor{red}{\textless logical\textgreater} tripleInteractionTimeIsSettable() Specify whether the \texttt{tripleInteractionTime} property of the \texttt{blackHole} component is settable.

tfamily \textcolor{red}{\textless void\textgreater} tripleInteractionTimeSet(<double> value) Set the \texttt{tripleInteractionTime} property of the \texttt{blackHole} component.

tfamily \textcolor{red}{\textless type(varying_string)\textgreater} type() Return the type of this object.

\begin{verbatim}
nodeComponentBlackHoleStandard

\textcolor{red}{\textless double\textgreater} accretionRate() Get the \texttt{accretionRate} property of the \texttt{blackHole} component.

\textcolor{red}{\textless type(varying_string)\textgreater} accretionRateAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the \texttt{accretionRate} property of the \texttt{blackHole} component.

\textcolor{red}{\textless void\textgreater} accretionRateFunction(<function()> deferredFunction) Set the function to be used for the \texttt{get} method of the \texttt{accretionRate} property of the \texttt{BlackHoleStandard} component.

\textcolor{red}{\textless logical\textgreater} accretionRateIsAttached() Return whether the \texttt{get} method of the \texttt{accretionRate} property of the \texttt{BlackHoleStandard} component has been attached to a function.

\textcolor{red}{\textless double\textgreater} accretionRateIsGettable() Get the \texttt{accretionRate} property of the \texttt{blackHole} component.

\textcolor{red}{\textless logical\textgreater} accretionRateIsSettable() Specify whether the \texttt{accretionRate} property of the \texttt{blackHole} component is settable.

\textcolor{red}{\textless void\textgreater} builder(<type(node)>componentDefinition →) Build a \texttt{nodeComponent} from a supplied XML definition.

\textcolor{red}{\textless double\textgreater} density(<double(3)> positionSpherical →, <componentType>[componentType] →, <massType>[massType] →, <weightBy>[weightBy] →, <integer>[weightIndex] →, <logical>[haloLoaded] →) Compute the density.

\textcolor{red}{\textless void\textgreater} deserializeRates(<double(:)> array ←) Deserialize the evolvable rates from an array.

\textcolor{red}{\textless void\textgreater} deserializeScales(<double(:)> array ←) Deserialize the evolvable scales from an array.

\textcolor{red}{\textless void\textgreater} deserializeValues(<double(:)> array ←) Deserialize the evolvable quantities from an array.

\textcolor{red}{\textless void\textgreater} destroy() Destroy the object.

\textcolor{red}{\textless void\textgreater} dump() Generate an ASCII dump of all properties.

\textcolor{red}{\textless void\textgreater} dumpRaw(<integer> fileHandle →) Generate a binary dump of all properties.

\textcolor{red}{\textless void\textgreater} dumpXML() Generate an XML dump of all properties.

\textcolor{red}{\textless double\textgreater} enclosedMass(<double> radius →, <componentType>[componentType] →, <massType>[massType] →, <weightBy>[weightBy] →, <integer>[weightIndex] →, <logical>[haloLoaded] →) Compute the mass enclosed within a radius.
\end{verbatim}
16.5. Objects

```
tfamily \textcolor{red}\{\textless *type(treeNode)\} \textgreater\} host() Return a pointer to the host
treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the
hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the
hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass
property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}\{\textless double\} \textgreater\} mass() Get the mass property of the blackHole
component.

tfamily \textcolor{red}\{\textless type(varying_string)\} \textgreater\} massAttributeMatch(<logical>[requireGettable]→,
<logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that
provide the given list off attributes for the mass property of the blackHole component.

tfamily \intzero massCount() Compute the count of evolvable quantities in the mass property of the
BlackHoleSimple component.

tfamily \textcolor{red}\{\textless double\} \textgreater\} massIsGettable() Get the mass property of the
blackHole component.

tfamily \logicalzero massIsSettable() Specify whether the mass property of the blackHole component
is settable.

tfamily \void massRate(<double> value) Cumulate to the rate of the mass property of the BlackHoleSimple
component.

tfamily \void massScale(<double> value) Set the scale of the mass property of the BlackHoleSimple
component.

tfamily \textcolor{red}\{\textless double\} \textgreater\} massSeed() Get the massSeed property of the
blackHole component.

tfamily \textcolor{red}\{\textless type(varying_string)\} \textgreater\} massSeedAttributeMatch(<logical>[requireGettable]→,
<logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that
provide the given list off attributes for the massSeed property of the blackHole component.

tfamily \textcolor{red}\{\textless double\} \textgreater\} massSeedIsGettable() Get the massSeed property
of the blackHole component.

tfamily \logicalzero massSeedIsSettable() Specify whether the massSeed property of the blackHole component
is settable.

tfamily \void massSet(<double> value) Set the mass property of the blackHole component.

tfamily \void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a
property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the blackHole component
class is active.
```
16. Coding Galacticus

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \void outputCount(<integer>integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \void outputNames(<integer>integerProperty ↔, <char[*](::)>integerPropertyNames ↔, <char[*](::)>integerPropertyComments ↔, <double(:)>integerPropertyUnitsSI ↔, <integer>doubleProperty ↔, <char[*](::)>doublePropertyNames ↔, <char[*](::)>doublePropertyComments ↔, <double(:)>doublePropertyUnitsSI ↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \double zero potential(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the gravitational potential.

tfamily \textcolor{red}{\textless double\textgreater} radialPosition() Get the radialPosition property of the blackHole component.

tfamily \textcolor{red}{\textless\textless type(varying\_string)(::)\textgreater} radialPositionAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the radialPosition property of the blackHole component

tfamily \int zero radialPositionCount() Compute the count of evolvable quantities in the radialPosition property of the BlackHoleStandard component.

tfamily \textcolor{red}{\textless double\textgreater} radialPositionIsGettable() Get the radialPosition property of the blackHole component.

tfamily \logical zero radialPositionIsSettable() Specify whether the radialPosition property of the blackHole component is settable.

tfamily \void radialPositionRate(<double> value) Cumulate to the rate of the radialPosition property of the BlackHoleStandard component.

tfamily \void radialPositionScale(<double> value) Set the scale of the radialPosition property of the BlackHoleStandard component.

tfamily \void radialPositionSet(<double> value) Set the radialPosition property of the blackHole component.

tfamily \textcolor{red}{\textless double\textgreater} radiativeEfficiency() Get the radiativeEfficiency property of the blackHole component.

tfamily \textcolor{red}{\textless\textless type(varying\_string)(::)\textgreater} radiativeEfficiencyAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the radiativeEfficiency property of the blackHole component

tfamily \void radiativeEfficiencyFunction(<function()> deferredFunction) Set the function to be used for the get method of the radiativeEfficiency property of the BlackHoleStandard component.
16.5. Objects

tfamily \logicalzero radiativeEfficiencyIsAttached() Return whether the get method of the radiativeEfficiency property of the BlackHoleStandard component has been attached to a function.

tfamily \textcolor{red}\textless\textgreater\textless\textgreater radiativeEfficiencyIsGettable() Get the radiativeEfficiency property of the blackHole component.

tfamily \logicalzero radiativeEfficiencyIsSettable() Specify whether the radiativeEfficiency property of the blackHole component is settable.

tfamily \void readRaw(<integer> fileHandle \mapsto) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezero rotationCurve(<double> radius \mapsto, <componentType> [componentType] \mapsto, <massType> [massType] \mapsto, <logical> [haloLoaded] \mapsto) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double> radius \mapsto, <componentType> [componentType] \mapsto, <massType> [massType] \mapsto, <logical> [haloLoaded] \mapsto) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array \mapsto) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array \mapsto) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array \mapsto) Serialize the evolvable quantities to an array.

tfamily \logicalzero simpleIsActive() Return whether the simple implementation of the blackHole component class is active.

tfamily \textcolor{red}\textless\textgreater\textless\textgreater spin() Get the spin property of the blackHole component.

tfamily \textcolor{red}\textless\textgreater spinAttributeMatch(<logical>[requireGettable] \mapsto, <logical>[requireSettable] \mapsto, <logical>[requireEvolvable] \mapsto) Return a list of implementations that provide the given list off attributes for the spin property of the blackHole component

tfamily \intzero spinCount() Compute the count of evolvable quantities in the spin property of the BlackHoleStandard component.

tfamily \textcolor{red}\textless\textgreater spinIsGettable() Get the spin property of the blackHole component.

tfamily \logicalzero spinIsSettable() Specify whether the spin property of the blackHole component is settable.

tfamily \void spinRate(<double> value) Cumulate to the rate of the spin property of the BlackHoleStandard component.

tfamily \void spinScale(<double> value) Set the scale of the spin property of the BlackHoleStandard component.

tfamily \textcolor{red}\textless\textgreater spinSeed() Get the spinSeed property of the blackHole component.

tfamily \textcolor{red}\textless\textgreater spinSeedAttributeMatch(<logical>[requireGettable] \mapsto, <logical>[requireSettable] \mapsto, <logical>[requireEvolvable] \mapsto) Return a list of implementations that provide the given list off attributes for the spinSeed property of the blackHole component
16. Coding GALACTICUS

tfamily \textcolor{red}{\textless \textbf{double}\textgreater} spinSeedIsGettable() Get the \textit{spinSeed} property of the \textit{blackHole} component.

tfamily \textcolor{red}{\textless \textbf{logical}\textgreater} spinSeedIsSettable() Specify whether the \textit{spinSeed} property of the \textit{blackHole} component is settable.

tfamily \textcolor{red}{\textbf{void}} spinSet(<\textbf{double}> value) Set the \textit{spin} property of the \textit{blackHole} component.

tfamily \textcolor{red}{\textless \textbf{logical}\textgreater} standardIsActive() Return whether the standard implementation of the \textit{blackHole} component class is active.

tfamily \textcolor{red}{\textbf{double}} surfaceDensity(<\textbf{double}(3)> positionCylindrical\textrightarrow{}, <\textit{componentType}> \textrightarrow{}, <\textit{massType}> \textrightarrow{}, <\textit{weightBy}> \textrightarrow{}, <\textbf{integer}> \textrightarrow{}, <\textbf{logical}> \textrightarrow{}, <\textbf{logical}> \textrightarrow{}) Compute the surface density.

tfamily \textcolor{red}{\textbf{void}} nodeComponentDarkMatterProfile

tfamily \textbf{void} builder(<\textbf{type(node)}\textrightarrow\textit{componentDefinition}) Build a \textit{nodeComponent} from a supplied XML definition.

tfamily \textbf{double} density(<\textbf{double}(3)> positionSpherical\textrightarrow{}, <\textit{componentType}> \textrightarrow{}, <\textit{massType}> \textrightarrow{}, <\textit{weightBy}> \textrightarrow{}, <\textbf{integer}> \textrightarrow{}, <\textbf{logical}> \textrightarrow{}, <\textbf{logical}> \textrightarrow{}) Compute the density.

tfamily \textbf{void} deserializeRates(<\textbf{double}(:)> array\textleft\textrightarrow) Deserialize the evolvable rates from an array.

tfamily \textbf{void} deserializeScales(<\textbf{double}(:)> array\textleft\textrightarrow) Deserialize the evolvable scales from an array.

tfamily \textbf{void} deserializeValues(<\textbf{double}(:)> array\textleft\textrightarrow) Deserialize the evolvable quantities from an array.

tfamily \textbf{void} destroy() Destroy the object.

tfamily \textbf{void} dump() Generate an ASCII dump of all properties.

tfamily \textbf{void} dumpRaw(<\textbf{integer}> fileHandle\textrightarrow) Generate a binary dump of all properties.
16.5. Objects

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(\double radius\rightarrow, \componentType\componentType\rightarrow, \massType\massType\rightarrow, \weightBy\weightBy\rightarrow, \integer\integer\rightarrow, \logical\logical\rightarrow) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless \type(treeNode)\textgreater} host() Return a pointer to the host \treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(\type(abundances) value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(\double value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(\double value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(\integer count\rightarrow, \varying_string name\leftarrow) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the darkMatterProfile component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(\integer integerProperty\leftrightarrow, \integer integerBufferCount\leftrightarrow, \integer(\ldots) integerBuffer\leftrightarrow, \integer doubleProperty\leftrightarrow, \integer doubleBufferCount\leftrightarrow, \double(\ldots) doubleBuffer\leftrightarrow, \double(\ldots) time\rightarrow, \integer instance\rightarrow) Generate values of outputtable properties.

tfamily \void outputCount(\integer integerPropertyCount\leftrightarrow, \integer doublePropertyCount\leftrightarrow, \double(\ldots) time\rightarrow, \integer instance\rightarrow) Compute a count of outputtable properties.

tfamily \void outputNames(\integer integerProperty\leftrightarrow, \char[^*][();] integerPropertyNames\leftrightarrow, \char[^*][();] integerPropertyComments\leftrightarrow, \double(\ldots) integerPropertyUnitsSI\leftrightarrow, \integer doubleProperty\leftrightarrow, \char[^*][();] doublePropertyNames\leftrightarrow, \char[^*][();] doublePropertyComments\leftrightarrow, \double(\ldots) doublePropertyUnitsSI\leftrightarrow, \double(\ldots) time\rightarrow, \integer instance\rightarrow) Generate names of outputtable properties.

tfamily \doublezero potential(\double radius\rightarrow, \componentType\componentType\rightarrow, \massType\massType\rightarrow, \logical\logical\rightarrow) Compute the gravitational potential.

tfamily \doublezero rotationCurve(\double radius\rightarrow, \componentType\componentType\rightarrow, \massType\massType\rightarrow, \logical\logical\rightarrow) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(\double radius\rightarrow, \componentType\componentType\rightarrow, \massType\massType\rightarrow, \logical\logical\rightarrow) Compute the rotation curve gradient.

tfamily \textcolor{red}{\textless \double\textgreater} scale() Get the scale property of the darkMatterProfile component.
tfamily \textcolor{red}{\textless type(varying\_string)(::)}\textgreater \textcolor{red}{\textless \textgreater} scaleAttributeMatch(\textcolor{red}{\textless logical\textgreater}\textcolor{red}{\textless requireGettable\textgreater}\textcolor{red}{\textless requireSettable\textgreater}\textcolor{red}{\textless requireEvolvable\textgreater}) Return a list of implementations that provide the given list of attributes for the scale property of the darkMatterProfile component.

tfamily \intzero scaleCount() Compute the count of evolvable quantities in the scale property of the DarkMatterProfileScale component.

tfamily \textcolor{red}{\textless text double\textgreater} scaleGrowthRate() Get the scaleGrowthRate property of the darkMatterProfile component.

tfamily \textcolor{red}{\textless type(varying\_string)(::)}\textgreater scaleGrowthRateAttributeMatch(\textcolor{red}{\textless logical\textgreater}\textcolor{red}{\textless requireGettable\textgreater}\textcolor{red}{\textless requireSettable\textgreater}\textcolor{red}{\textless requireEvolvable\textgreater}) Return a list of implementations that provide the given list of attributes for the scaleGrowthRate property of the darkMatterProfile component.

tfamily \textcolor{red}{\textless double\textgreater} scaleGrowthRateIsGettable() Get the scaleGrowthRate property of the darkMatterProfile component.

tfamily \logicalzero scaleGrowthRateIsSettable() Specify whether the scaleGrowthRate property of the darkMatterProfile component is settable.

tfamily \void scaleGrowthRateSet(\textcolor{red}{\textless double\textgreater} value) Set the scaleGrowthRate property of the darkMatterProfile component.

tfamily \logicalzero scaleIsActive() Return whether the scale implementation of the darkMatterProfile component class is active.

tfamily \textcolor{red}{\textless double\textgreater} scaleIsGettable() Get the scale property of the darkMatterProfile component.

tfamily \logicalzero scaleIsSettable() Specify whether the scale property of the darkMatterProfile component is settable.

tfamily \logicalzero scalePresetIsActive() Return whether the scalePreset implementation of the darkMatterProfile component class is active.

tfamily \void scaleRate(\textcolor{red}{\textless double\textgreater} value) Cumulate to the rate of the scale property of the DarkMatterProfileScale component.

tfamily \void scaleScale(\textcolor{red}{\textless double\textgreater} value) Set the scale of the scale property of the DarkMatterProfileScale component.

tfamily \void scaleSet(\textcolor{red}{\textless double\textgreater} value) Set the scale property of the darkMatterProfile component.

tfamily \logicalzero scaleShapeIsActive() Return whether the scaleShape implementation of the darkMatterProfile component class is active.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(\textcolor{red}{\textless double(:)}\textgreater\textcolor{red}{\textless array\textgreater}) Serialize the evolvable rates to an array.

tfamily \void serializeScales(\textcolor{red}{\textless double(:)}\textgreater\textcolor{red}{\textless array\textgreater}) Serialize the evolvable scales to an array.

tfamily \void serializeValues(\textcolor{red}{\textless double(:)}\textgreater\textcolor{red}{\textless array\textgreater}) Serialize the evolvable quantities to an array.

tfamily \textcolor{red}{\textless double\textgreater} shape() Get the shape property of the darkMatterProfile component.
16.5. Objects

tfamily \textcolor{red}{\textless type(varying\_string):(\textgreater\textgreater} shapeAttributeMatch(\textless logical:\textgreater[requireGettable]→, \textless logical:\textgreater[requireSettable]→, \textless logical:\textgreater[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the shape property of the darkMatterProfile component

tfamily \textcolor{red}{\textless double\textgreater} shapeCount() Compute the count of evolvable quantities in the shape property of the DarkMatterProfileScaleShape component.

tfamily \textcolor{red}{\textless double\textgreater} shapeGrowthRate() Get the shapeGrowthRate property of the darkMatterProfile component.

tfamily \textcolor{red}{\textless type(varying\_string):(\textgreater\textgreater} shapeGrowthRateAttributeMatch(\textless logical:\textgreater[requireGettable]→, \textless logical:\textgreater[requireSettable]→, \textless logical:\textgreater[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the shapeGrowthRate property of the darkMatterProfile component

tfamily \textcolor{red}{\textless double\textgreater} shapeGrowthRateIsGettable() Get the shapeGrowthRate property of the darkMatterProfile component.

tfamily \logicalzero shapeGrowthRateIsSettable() Specify whether the shapeGrowthRate property of the darkMatterProfile component is settable.

tfamily \void shapeGrowthRateSet(<double> value) Set the shapeGrowthRate property of the darkMatterProfile component.

tfamily \textcolor{red}{\textless double\textgreater} shapeIsGettable() Get the shape property of the darkMatterProfile component.

tfamily \logicalzero shapeIsSettable() Specify whether the shape property of the darkMatterProfile component is settable.

tfamily \void shapeRate(<double> value) Cumulateto the rate of the shape property of the DarkMatterProfileScaleShape component.

tfamily \void shapeScale(<double> value) Set the scale of the shape property of the DarkMatterProfileScaleShape component.

tfamily \void shapeSet(<double> value) Set the shape property of the darkMatterProfile component.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical→, \textless componentType\textgreater[componentType]→, \textless massType\textgreater[massType]→, \textless weightBy\textgreater[weightBy]→, \textless integer\textgreater[weightIndex]→, \textless logical\textgreater[haloLoaded]→) Compute the surface density.

tfamily \textcolor{red}{\textless type(varying\_string):(\textgreater\textgreater} type() Return the type of this object.

tfamily nodeComponentDarkMatterProfileNull

tfamily \void builder(<\*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily \doublezero density(<double(3)> positionSpherical→, \textless componentType\textgreater[componentType]→, \textless massType\textgreater[massType]→, \textless weightBy\textgreater[weightBy]→, \textless integer\textgreater[weightIndex]→, \textless logical\textgreater[haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

429
tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \double zero enclosedMass(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→)
Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

tfamily \logical zero nullIsActive() Return whether the null implementation of the darkMatterProfile component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \void outputCount(<integer>integerPropertyCount↔, <integer>doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \void outputNames(<integer>integerProperty↔, <char[*](::)> integerPropertyNames↔, <char[*]::)> integerPropertyComments↔, <double::> integerPropertyUnitsSI↔, <integer>doubleProperty↔, <char[*]::)> doublePropertyNames↔, <char[*]::)> doublePropertyComments↔, <double::> doublePropertyUnitsSI↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \double zero potential(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the gravitational potential.
16.5. Objects

tfamily \textcolor{red}{\textless double\textgreater} scale() Get the scale property of the darkMatterProfile component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater scaleGrowthRate() Get the scaleGrowthRate property of the darkMatterProfile component.

tfamily \textcolor{red}{\textless double\textgreater} scaleGrowthRateIsGettable() Get the scaleGrowthRate property of the darkMatterProfile component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater scaleGrowthRateIsSettable() Specify whether the scaleGrowthRate property of the darkMatterProfile component is settable.

tfamily \void scaleGrowthRateSet(<double> value) Set the scaleGrowthRate property of the darkMatterProfile component.

tfamily \logicalzero scaleIsActive() Return whether the scale implementation of the darkMatterProfile component class is active.

tfamily \textcolor{red}{\textless double\textgreater} scaleIsGettable() Get the scale property of the darkMatterProfile component.

tfamily \logicalzero scaleIsSettable() Specify whether the scale property of the darkMatterProfile component is settable.

tfamily \logicalzero scalePresetIsActive() Return whether the scalePreset implementation of the darkMatterProfile component class is active.

tfamily \void scaleRate(<double> value) Cumulate to the rate of the scale property of the darkMatterProfileScale component.

tfamily \void scaleScale(<double> value) Set the scale of the scale property of the darkMatterProfileScale component.

tfamily \void scaleSet(<double> value) Set the scale property of the darkMatterProfile component.
tfamily \logicalzero scaleShapeIsActive() Return whether the scaleShape implementation of the dark-MatterProfile component class is active.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array) Serialize the evolvable quantities to an array.

tfamily \textcolor{red}\textless\textgreater shape() Get the shape property of the darkMatterProfile component.

tfamily \textcolor{red}\textless type(varying\_string)(:)\textgreater shapeAttributeMatch(<logical>[requireGettable]→ <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the shape property of the darkMatterProfile component

tfamily \intzero shapeCount() Compute the count of evolvable quantities in the shape property of the DarkMatterProfileScaleShape component.

tfamily \textcolor{red}\textless double\textgreater shapeGrowthRate() Get the shapeGrowthRate property of the darkMatterProfile component.

tfamily \textcolor{red}\textless type(varying\_string)(:)\textgreater shapeGrowthRateAttributeMatch(<logical>[requireGettable]→ <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the shapeGrowthRate property of the darkMatterProfile component

tfamily \textcolor{red}\textless double\textgreater shapeGrowthRateIsGettable() Get the shapeGrowthRate property of the darkMatterProfile component.

tfamily \logicalzero shapeGrowthRateIsSettable() Specify whether the shapeGrowthRate property of the darkMatterProfile component is settable.

tfamily \void shapeGrowthRateSet(<double> value) Set the shapeGrowthRate property of the darkMatterProfile component.

tfamily \textcolor{red}\textless double\textgreater shapeIsGettable() Get the shape property of the darkMatterProfile component.

tfamily \logicalzero shapeIsSettable() Specify whether the shape property of the darkMatterProfile component is settable.

tfamily \void shapeRate(<double> value) Cumulate to the rate of the shape property of the DarkMatterProfileScaleShape component.

tfamily \void shapeScale(<double> value) Set the scale of the shape property of the DarkMatterProfileScaleShape component.

tfamily \void shapeSet(<double> value) Set the shape property of the darkMatterProfile component.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.

tfamily \textcolor{red}\textless type(varying\_string)\textgreater type() Return the type of this object.
16.5. Objects

```cpp
// tfamily nodeComponentDarkMatterProfileScale

// tfamily \void builder(<\texttt{*type(node)}\texttt{>componentDefinition→}) Build a nodeComponent from a supplied XML definition.

// tfamily \double zero density(<\texttt{double(3)}→positionSpherical→, <\texttt{componentType} [componentType→), <\texttt{massType} [massType→, <\texttt{weightBy} [weightBy→, <\texttt{integer} [weightIndex→, <\texttt{logical} [haloLoaded→) Compute the density.

// tfamily \void deserializeRates(<\texttt{double(:)}→array←) Deserialize the evolvable rates from an array.

// tfamily \void deserializeScales(<\texttt{double(:)}→array←) Deserialize the evolvable scales from an array.

// tfamily \void deserializeValues(<\texttt{double(:)}→array←) Deserialize the evolvable quantities from an array.

// tfamily \void destroy() Destroy the object.

// tfamily \void dump() Generate an ASCII dump of all properties.

// tfamily \void dumpRaw(<\texttt{integer}→fileHandle←) Generate a binary dump of all properties.

// tfamily \void dumpXML() Generate an XML dump of all properties.

// tfamily \double zero enclosedMass(<\texttt{double}→radius→, <\texttt{componentType} [componentType→), <\texttt{massType} [massType→, <\texttt{weightBy} [weightBy→, <\texttt{integer} [weightIndex→, <\texttt{logical} [haloLoaded→) Compute the mass enclosed within a radius.

// tfamily \textcolor{red}{\texttt{\textbackslash{}}textless \texttt{*type(treeNode)}\texttt{\textgreater}host() Return a pointer to the host treeNode object.

// tfamily \void hotHaloCoolingAbundancesRate(<\texttt{type(abundances)}→value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

// tfamily \void hotHaloCoolingAngularMomentumRate(<\texttt{double}→value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

// tfamily \void hotHaloCoolingMassRate(<\texttt{double}→value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

// tfamily \void initialize() Initialize the object.

// tfamily \void nameFromIndex(<\texttt{integer}→count→, <\texttt{varying_string}→name←) Return the name of a property given is index.

// tfamily \logical zero nullIsActive() Return whether the null implementation of the darkMatterProfile component class is active.

// tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

// tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

// tfamily \void output(<\texttt{integer}→integerProperty←, <\texttt{integer}→integerBufferCount←, <\texttt{integer(:,:)}→integerBuffer←, <\texttt{integer}→doubleProperty←, <\texttt{integer}→doubleBufferCount←, <\texttt{double(:,:)}→doubleBuffer←, <\texttt{double}→time→, <\texttt{integer}→instance←) Generate values of outputtable properties.

// tfamily \void outputCount(<\texttt{integer}→integerPropertyCount←, <\texttt{integer}→doublePropertyCount←, <\texttt{double}→time→, <\texttt{integer}→instance←) Compute a count of outputtable properties.
```
16. Coding Galacticus

tfamily \( \text{\textcolor{red}{\textgreater double\textless}} \text{scale()} \) Get the \text{scale} property of the \text{darkMatterProfile} component.

tfamily \( \text{\textcolor{red}{\textless type(varying\_\textgreater\_string)(:)}\textgreater\textcolor{red}{\textless double\textgreater}} \text{scaleGrowthRate()} \) Get the \text{scaleGrowthRate} property of the \text{darkMatterProfile} component.

tfamily \( \text{\textcolor{red}{\textless double\textgreater}} \text{scaleGrowthRateIsGettable()} \) Get the \text{scaleGrowthRate} property of the \text{darkMatterProfile} component.

tfamily \( \text{\textcolor{red}{\textless double\textgreater}} \text{scaleGrowthRateIsSettable()} \) Specify whether the \text{scaleGrowthRate} property of the \text{darkMatterProfile} component is settable.

tfamily \( \text{\textcolor{red}{\textless double\textgreater}} \text{scaleIsActive()} \) Return whether the scale implementation of the \text{darkMatterProfile} component class is active.

tfamily \( \text{\textcolor{red}{\textless double\textgreater}} \text{scaleIsAttached()} \) Return whether the get method of the scale property of the \text{DarkMatterProfileScale} component has been attached to a function.

tfamily \( \text{\textcolor{red}{\textless double\textgreater}} \text{scaleIsGettable()} \) Get the \text{scale} property of the \text{darkMatterProfile} component.
16.5. Objects

tfamily \logicalzero scalesIsSettable() Specify whether the scale property of the darkMatterProfile component is settable.

tfamily \logicalzero scalePresetIsActive() Return whether the scalePreset implementation of the darkMatterProfile component class is active.

tfamily \void scaleRate(<double> value) Cumulate to the rate of the scale property of the DarkMatterProfileScale component.

tfamily \void scaleScale(<double> value) Set the scale of the scale property of the DarkMatterProfileScale component.

tfamily \void scaleSet(<double> value) Set the scale property of the darkMatterProfile component.

tfamily \logicalzero scaleShapeIsActive() Return whether the scaleShape implementation of the darkMatterProfile component class is active.

tfamily \textcolor{red}{\lessdouble\textgreater} scaleValue() Get the scale property of the darkMatterProfile component.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array) Serialize the evolvable quantities to an array.

tfamily \textcolor{red}{\lessdouble\textgreater} shape() Get the shape property of the darkMatterProfile component.

tfamily \textcolor{red}{\less type(varying_string)(:)} shapeAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the shape property of the darkMatterProfile component.

tfamily \intzero shapeCount() Compute the count of evolvable quantities in the shape property of the DarkMatterProfileScaleShape component.

tfamily \textcolor{red}{\less double\textgreater} shapeGrowthRate() Get the shapeGrowthRate property of the darkMatterProfile component.

tfamily \textcolor{red}{\less type(varying_string)(:)} shapeGrowthRateAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the shapeGrowthRate property of the darkMatterProfile component.

tfamily \textcolor{red}{\less double\textgreater} shapeGrowthRateIsGettable() Get the shapeGrowthRate property of the darkMatterProfile component.

tfamily \logicalzero shapeGrowthRateIsSettable() Specify whether the shapeGrowthRate property of the darkMatterProfile component is settable.

tfamily \void shapeGrowthRateSet(<double> value) Set the shapeGrowthRate property of the darkMatterProfile component.

tfamily \textcolor{red}{\less double\textgreater} shapeIsGettable() Get the shape property of the darkMatterProfile component.
tfamily \logicalzero shapeIsSettable() Specify whether the shape property of the darkMatterProfile component is settable.

tfamily \void shapeRate(<double> value) Cumulate to the rate of the shape property of the DarkMatterProfileScaleShape component.

tfamily \void shapeScale(<double> value) Set the scale of the shape property of the DarkMatterProfileScaleShape component.

tfamily \void shapeSet(<double> value) Set the shape property of the darkMatterProfile component.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

tfamily nodeComponentDarkMatterProfileScalePreset

tfamily \void builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily \doublezero density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.
16.5. Objects

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count →, <varying_string> name ←) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the darkMatterProfile component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:) > integerBuffer ↔, <integer> doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:) > doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty ↔, <char[*](:)> integerPropertyNames ↔, <char[*](:)> integerPropertyComments ↔, <double(:)> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char[*](:)> doublePropertyNames ↔, <char[*](:)> doublePropertyComments ↔, <double(:)> doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

tfamily \doublezero potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

tfamily \void readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezero rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

tfamily \textcolor{red}{\textless double\textgreater} scale() Get the scale property of the darkMatterProfile component.

tfamily \textcolor{red}{\textless type(varying_string)(:)\textgreater} scaleAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the scale property of the darkMatterProfile component

tfamily \intzero scaleCount() Compute the count of evolvable quantities in the scale property of the DarkMatterProfileScale component.

tfamily \textcolor{red}{\textless double\textgreater} scaleGrowthRate() Get the scaleGrowthRate property of the darkMatterProfile component.

tfamily \textcolor{red}{\textless type(varying_string)(:)\textgreater} scaleGrowthRateAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the scaleGrowthRate property of the darkMatterProfile component

437
tfamily \textcolor{red}{\textless double\textgreater} scaleGrowthRateIsGettable() Get the \texttt{scaleGrowthRate} property of the \texttt{darkMatterProfile} component.

tfamily \textcolor{red}{\textless double\textgreater} scaleGrowthRateIsSettable() Specify whether the \texttt{scaleGrowthRate} property of the \texttt{darkMatterProfile} component is settable.

tfamily \void scaleGrowthRateSet(<double> value) Set the \texttt{scaleGrowthRate} property of the \texttt{darkMatterProfile} component.

tfamily \logicalzero scaleIsActive() Return whether the scale implementation of the \texttt{darkMatterProfile} component class is active.

tfamily \textcolor{red}{\textless double\textgreater} scaleIsGettable() Get the \texttt{scale} property of the \texttt{darkMatterProfile} component.

tfamily \logicalzero scaleIsSettable() Specify whether the \texttt{scale} property of the \texttt{darkMatterProfile} component is settable.

tfamily \logicalzero scalePresetIsActive() Return whether the scalePreset implementation of the dark-MatterProfile component class is active.

tfamily \void scaleRate(<double> value) Cumulate to the rate of the \texttt{scale} property of the \texttt{DarkMatterProfileScale} component.

tfamily \void scaleScale(<double> value) Set the scale of the \texttt{scale} property of the \texttt{DarkMatterProfileScale} component.

tfamily \void scaleSet(<double> value) Set the \texttt{scale} property of the \texttt{darkMatterProfile} component.

tfamily \logicalzero scaleShapeIsActive() Return whether the scaleShape implementation of the \texttt{darkMatterProfile} component class is active.

tfamily \intzero serializeCount()Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

tfamily \textcolor{red}{\textless double\textgreater} shape() Get the \texttt{shape} property of the \texttt{darkMatterProfile} component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater shapeAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the \texttt{shape} property of the \texttt{darkMatterProfile} component.

tfamily \intzero shapeCount() Compute the count of evolvable quantities in the \texttt{shape} property of the \texttt{DarkMatterProfileScaleShape} component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} shapeGrowthRate() Get the \texttt{shapeGrowthRate} property of the \texttt{darkMatterProfile} component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} shapeGrowthRateAttributeM-

[438]atch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the \texttt{shapeGrowthRate} property of the \texttt{darkMatterProfile} component.
16.5. Objects

tfamily \textcolor{red}{\langle double\rangle} shapeGrowthRateIsGettable() Get the \texttt{shapeGrowthRate} property of the \texttt{darkMatterProfile} component.

tfamily \texttt{logical zero} shapeGrowthRateIsSettable() Specify whether the \texttt{shapeGrowthRate} property of the \texttt{darkMatterProfile} component is settable.

tfamily \texttt{void} shapeGrowthRateSet(<double> value) Set the \texttt{shapeGrowthRate} property of the \texttt{darkMatterProfile} component.

tfamily \textcolor{red}{\langle double\rangle} shapeIsGettable() Get the \texttt{shape} property of the \texttt{darkMatterProfile} component.

tfamily \texttt{logical zero} shapeIsSettable() Specify whether the \texttt{shape} property of the \texttt{darkMatterProfile} component is settable.

tfamily \texttt{void} shapeRate(<double> value) Cumulate to the rate of the \texttt{shape} property of the \texttt{DarkMatterProfileScaleShape} component.

tfamily \texttt{void} shapeScale(<double> value) Set the scale of the \texttt{shape} property of the \texttt{DarkMatterProfileScaleShape} component.

tfamily \texttt{void} shapeSet(<double> value) Set the \texttt{shape} property of the \texttt{darkMatterProfile} component.

tfamily \texttt{double zero} surfaceDensity(<double(3)> positionCylindrical\rightarrow, <componentType> \rightarrow, <massType> \rightarrow, <weightBy> \rightarrow, <integer> \rightarrow, \texttt{logical} \rightarrow) Compute the surface density.

tfamily \textcolor{red}{\langle type(varying\_string)\rangle} type() Return the type of this object.

tfamily nodeComponentDarkMatterProfileScaleShape

tfamily \texttt{void} builder(<\texttt{type(node)}\rangle componentDefinition\rightarrow) Build a \texttt{nodeComponent} from a supplied XML definition.

tfamily \texttt{double zero} density(<double(3)> positionSpherical\rightarrow, <componentType> \rightarrow, <massType> \rightarrow, <weightBy> \rightarrow, <integer> \rightarrow, \texttt{logical} \rightarrow) Compute the density.

tfamily \texttt{void} deserializeRates(<double(:)> array\leftarrow) Deserialize the evolvable rates from an array.

tfamily \texttt{void} deserializeScales(<double(:)> array\leftarrow) Deserialize the evolvable scales from an array.

tfamily \texttt{void} deserializeValues(<double(:)> array\leftarrow) Deserialize the evolvable quantities from an array.

tfamily \texttt{void} destroy() Destroy the object.

tfamily \texttt{void} dump() Generate an ASCII dump of all properties.

tfamily \texttt{void} dumpRaw(<integer> fileHandle\rightarrow) Generate a binary dump of all properties.

tfamily \texttt{void} dumpXML() Generate an XML dump of all properties.

tfamily \texttt{double zero} enclosedMass(<double> radius\rightarrow, <componentType> \rightarrow, <massType> \rightarrow, <weightBy> \rightarrow, <integer> \rightarrow, \texttt{logical} \rightarrow) Compute the mass enclosed within a radius.
tfamily `{textcolor{red}{\textless type(treeNode)\textgreater}} host() Return a pointer to the host
  treeNode object.

tfamily `{textcolor{red}{\textless type(abundances)\textgreater}} hostHaloCoolingAbundancesRate(value) Cumulate to the rate of the
galacticus\text{hotHaloCoolingAbundances} property of the hotHalo component.

tfamily `{textcolor{red}{\textless double\textgreater}} hostHaloCoolingAngularMomentumRate(value) Cumulate to the rate of the
galacticus\text{hotHaloCoolingAngularMomentum} property of the hotHalo component.

tfamily `{textcolor{red}{\textless double\textgreater}} hostHaloCoolingMassRate(value) Cumulate to the rate of the galacticus\text{hotHaloCoolingMass} property of the hotHalo component.

tfamily initialize() Initialize the object.

tfamily nameFromIndex(count →, name ←) Return the name of a property given is index.

tfamily logicalzero nullIsActive() Return whether the null implementation of the darkMatterProfile component class is active.

tfamily odeStepRatesInitialize() Initialize rates for evolvable properties.

```
tfamily odeStepScalesInitialize() Initialize scales for evolvable properties.
```

tfamily output(integer integerProperty ↔, integerBufferCount ↔, doubleBuffer ↔,
  integerProperty count →, integerBufferCount →, doubleBuffer →, double time →, instance →) Generate values of outputtable properties.

```
tfamily outputCount(integer integerPropertyCount ↔, doublePropertyCount ↔,
  double time →, instance →) Compute a count of outputtable properties.
```

```
tfamily outputNames(integerProperty ↔, integerPropertyNames ↔, integerPropertyComments ↔,
  integerPropertyUnitsSI ↔, doubleProperty ↔, doublePropertyNames ↔, doublePropertyComments ↔,
  doublePropertyUnitsSI ↔, double time →, instance →) Generate names of outputtable properties.
```

```
tfamily double potential(radius →, componentType [componentType] →, massType [massType] →, logical [haloLoaded] →) Compute the gravitational potential.
```

```
tfamily readRaw(fileHandle →) Read a binary dump of the nodeComponent from the
given fileHandle.
```

```
tfamily double rotationCurve(radius →, componentType [componentType] →, massType [massType] →, logical [haloLoaded] →) Compute the rotation curve.
```

```
tfamily double rotationCurveGradient(radius →, componentType [componentType] →, massType [massType] →, logical [haloLoaded] →) Compute the rotation curve gradient.
```

```
tfamily `{textcolor{red}{\textless double\textgreater}} scale() Get the scale property of the darkMatterProfile component.
```

```
tfamily `{textcolor{red}{\textless type(varying_string)\textgreater}} scaleAttributeMatch(requireGettable →, requireSettable →, requireEvolvable →) Return a list of implementations that
  provide the given list off attributes for the scale property of the darkMatterProfile component.
```
16.5. Objects

tfamily intzero scaleCount() Compute the count of evolvable quantities in the scale property of the
DarkMatterProfileScale component.

tfamily void scaleFunction(<function()> deferredFunction) Set the function to be used for the get
method of the scale property of the DarkMatterProfileScale component.

tfamily textcolor{red}{\textless double\textgreater} scaleGrowthRate() Get the scaleGrowthRate
property of the DarkMatterProfile component.

tfamily textcolor{red}{\textless type(varying\_string)(:)} scaleGrowthRateAttributeMatch(<logical>|requireGettable|→, <logical>|requireSettable|→, <logical>|requireEvolvable|→) Return a list of implementations that provide the given list of attributes for the scaleGrowthRate
property of the darkMatterProfile component.

tfamily textcolor{red}{\textless double\textgreater} scaleGrowthRateIsGettable() Get the scaleGrowthRateIsGettable() Get the scaleGrowthRate
property of the darkMatterProfile component.

tfamily logicalzero scaleGrowthRateIsSettable() Specify whether the scaleGrowthRate property of the
darkMatterProfile component is settable.

tfamily void scaleGrowthRateSet(<double> value) Set the scaleGrowthRate property of the darkMatterProfile
component.

tfamily logicalzero scaleIsActive() Return whether the scale implementation of the darkMatterProfile
component class is active.

tfamily logicalzero scaleIsAttached() Return whether the get method of the scale property of the
DarkMatterProfileScale component has been attached to a function.

tfamily textcolor{red}{\textless double\textgreater} scaleIsGettable() Get the scale property of the
darkMatterProfile component.

tfamily logicalzero scaleIsSettable() Specify whether the scale property of the darkMatterProfile
component is settable.

tfamily logicalzero scalePresetIsActive() Return whether the scalePreset implementation of the darkMatterProfile component class is active.

tfamily void scaleRate(<double> value) Cumulate to the rate of the scale property of the DarkMatterProfileScale
component.

tfamily void scaleScale(<double> value) Set the scale of the scale property of the DarkMatterProfileScale
component.

tfamily void scaleSet(<double> value) Set the scale property of the darkMatterProfile component.

tfamily logicalzero scaleShapeIsActive() Return whether the scaleShape implementation of the darkMatterProfile component class is active.

tfamily textcolor{red}{\textless double\textgreater} scaleValue() Get the scale property of the darkMatterProfile
component.

tfamily intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily void serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.
tfamily \void serializeScales(\texttt{<double(:)> array}) Serialize the evolvable scales to an array.

tfamily \void serializeValues(\texttt{<double(:)> array}) Serialize the evolvable quantities to an array.

tfamily \textcolor{red}{\textless \texttt{double\textgreater}} \texttt{shape()} Get the \texttt{shape} property of the \texttt{darkMatterProfile} component.

tfamily \textcolor{red}{\textless \texttt{type(varying\_string):\textgreater}} \texttt{shapeAttributeMatch(\texttt{<logical>[requireGettable]→, \texttt{<logical>[requireSettable]→, \texttt{<logical>[requireEvolvable]→)}})} Return a list of implementations that provide the given list of attributes for the \texttt{shape} property of the \texttt{darkMatterProfile} component.

tfamily \intzero \texttt{shapeCount()} Compute the count of evolvable quantities in the \texttt{shape} property of the \texttt{DarkMatterProfileScaleShape} component.

tfamily \textcolor{red}{\textless \texttt{double\textgreater}} \texttt{shapeGrowthRate()} Get the \texttt{shapeGrowthRate} property of the \texttt{darkMatterProfile} component.

tfamily \textcolor{red}{\textless \texttt{type(varying\_string):\textgreater}} \texttt{shapeGrowthRateAttributeMatch(\texttt{<logical>[requireGettable]→, \texttt{<logical>[requireSettable]→, \texttt{<logical>[requireEvolvable]→)}})} Return a list of implementations that provide the given list of attributes for the \texttt{shapeGrowthRate} property of the \texttt{darkMatterProfile} component.

tfamily \textcolor{red}{\textless \texttt{double\textgreater}} \texttt{shapeGrowthRateIsGettable()} Get the \texttt{shapeGrowthRate} property of the \texttt{darkMatterProfile} component.

tfamily \logicalzero \texttt{shapeGrowthRateIsSettable()} Specify whether the \texttt{shapeGrowthRate} property of the \texttt{darkMatterProfile} component is settable.

tfamily \void \texttt{shapeGrowthRateSet(\texttt{<double> value})} Set the \texttt{shapeGrowthRate} property of the \texttt{darkMatterProfile} component.

tfamily \textcolor{red}{\textless \texttt{double\textgreater}} \texttt{shapeIsGettable()} Get the \texttt{shape} property of the \texttt{darkMatterProfile} component.

tfamily \logicalzero \texttt{shapeIsSettable()} Specify whether the \texttt{shape} property of the \texttt{darkMatterProfile} component is settable.

tfamily \void \texttt{shapeRate(\texttt{<double> value})} Cumulate to the rate of the \texttt{shape} property of the \texttt{DarkMatterProfileScaleShape} component.

tfamily \void \texttt{shapeScale(\texttt{<double> value})} Set the scale of the \texttt{shape} property of the \texttt{DarkMatterProfileScaleShape} component.

tfamily \void \texttt{shapeSet(\texttt{<double> value})} Set the \texttt{shape} property of the \texttt{darkMatterProfile} component.

tfamily \doublezero \texttt{surfaceDensity(\texttt{<double(3)> positionCylindrical→, \texttt{<componentType>[componentType]→, \texttt{<massType>[massType]→, \texttt{<weightBy>[weightBy]→, \texttt{<integer>[weightIndex]→, \texttt{<logical>[haloLoaded]→)}})}})} Compute the surface density.

tfamily \textcolor{red}{\textless \texttt{type(varying\_string):\textgreater}} \texttt{type()} Return the type of this object.
16.5. Objects

tfamily nodeComponentDisk

tfamily \textcolor{red}{\textless type(\textless double\textgreater)\textgreater} angularMomentum() Get the angularMomentum property of the disk component.

tfamily \textcolor{red}{\textless type(\textless double\textgreater)\textgreater} abundancesStellar() Get the abundancesStellar property of the disk component.

tfamily \textcolor{red}{\textless type(varying\_string)()\textgreater} abundancesStellarAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the abundancesStellar property of the disk component.

tfamily \intzero abundancesStellarCount() Compute the count of evolvable quantities in the abundancesStellar property of the DiskExponential component.

tfamily \textcolor{red}{\textless type(\textless double\textgreater)\textgreater} abundancesStellarIsGettable() Get the abundancesStellar property of the disk component.

tfamily \logicalzero abundancesStellarIsSettable() Specify whether the abundancesStellar property of the disk component is settable.

tfamily \void abundancesStellarRate(<type(abundances)> value) Cumulate to the rate of the abundancesStellar property of the DiskExponential component.

tfamily \void abundancesStellarScale(<type(abundances)> value) Set the scale of the abundancesStellar property of the DiskExponential component.

tfamily \void abundancesStellarSet(<type(abundances)> value) Set the abundancesStellar property of the disk component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesStellar() Get the abundancesStellar property of the disk component.

tfamily \textcolor{red}{\textless type(varying\_string)()\textgreater} abundancesStellarAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the abundancesStellar property of the disk component.

tfamily \intzero abundancesStellarCount() Compute the count of evolvable quantities in the abundancesStellar property of the DiskExponential component.

tfamily \textcolor{red}{\textless type(\textless double\textgreater)\textgreater} abundancesStellarIsGettable() Get the abundancesStellar property of the disk component.

tfamily \logicalzero abundancesStellarIsSettable() Specify whether the abundancesStellar property of the disk component is settable.

tfamily \void abundancesStellarRate(<type(abundances)> value) Cumulate to the rate of the abundancesStellar property of the DiskExponential component.

tfamily \void abundancesStellarScale(<type(abundances)> value) Set the scale of the abundancesStellar property of the DiskExponential component.

tfamily \void abundancesStellarSet(<type(abundances)> value) Set the abundancesStellar property of the disk component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesGas() Get the abundancesGas property of the disk component.

tfamily \textcolor{red}{\textless type(varying\_string)()\textgreater} abundancesGasAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the abundancesGas property of the disk component.

tfamily \intzero abundancesGasCount() Compute the count of evolvable quantities in the abundancesGas property of the DiskExponential component.

tfamily \textcolor{red}{\textless type(\textless double\textgreater)\textgreater} abundancesGasIsGettable() Get the abundancesGas property of the disk component.

tfamily \logicalzero abundancesGasIsSettable() Specify whether the abundancesGas property of the disk component is settable.

tfamily \void abundancesGasRate(<type(abundances)> value) Cumulate to the rate of the abundancesGas property of the DiskExponential component.

tfamily \void abundancesGasScale(<type(abundances)> value) Set the scale of the abundancesGas property of the DiskExponential component.

tfamily \void abundancesGasSet(<type(abundances)> value) Set the abundancesGas property of the disk component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesGas() Get the abundancesGas property of the disk component.

tfamily \textcolor{red}{\textless type(varying\_string)()\textgreater} abundancesGasAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the abundancesGas property of the disk component.
tfamily \textcolor{red}{\{\textless double\textgreater\}} angularMomentumAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the angularMomentum property of the disk component.

tfamily \intzero angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the DiskExponential component.

tfamily \textcolor{red}{\{\textless double\textgreater\}} angularMomentumIsGettable() Get the angularMomentum property of the disk component.

tfamily \logicalzero angularMomentumIsSettable() Specify whether the angularMomentum property of the disk component is settable.

tfamily \void angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the DiskExponential component.

tfamily \void angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the DiskExponential component.

tfamily \void angularMomentumSet(<double> value) Set the angularMomentum property of the disk component.

tfamily \void builder(<\*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily \doublezero density(<double(3)> positionSpherical→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<\integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \logicalzero exponentialIsActive() Return whether the exponential implementation of the disk component class is active.

tfamily \textcolor{red}{\{\textless double\textgreater\}} halfMassRadius() Get the halfMassRadius property of the disk component.

tfamily \textcolor{red}{\{\textless double\textgreater\}} halfMassRadiusAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the halfMassRadius property of the disk component.
16.5. Objects

tfamily \textcolor{red}{\textless double\textgreater} halfMassRadiusIsGettable() Get the halfMassRadius property of the disk component.

tfamily \textcolor{red}{\textless logical\textgreater} halfMassRadiusIsSettable() Specify whether the halfMassRadius property of the disk component is settable.

tfamily \textcolor{red}{\textless type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \textcolor{red}{\textless logical\textgreater} isInitialized() Get the isInitialized property of the disk component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} isInitializedAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the isInitialized property of the disk component.

tfamily \textcolor{red}{\textless logical\textgreater} isInitializedIsGettable() Get the isInitialized property of the disk component.

tfamily \textcolor{red}{\textless logical\textgreater} isInitializedIsSettable() Specify whether the isInitialized property of the disk component is settable.

tfamily \textcolor{red}{\textless logical\textgreater} isInitializedSet(<logical> value) Set the isInitialized property of the disk component.

tfamily \textcolor{red}{\textless type(stellarLuminosities)\textgreater} luminositiesStellar() Get the luminositiesStellar property of the disk component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} luminositiesStellarAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the luminositiesStellar property of the disk component.

tfamily \intzero luminositiesStellarCount() Compute the count of evolvable quantities in the luminositiesStellar property of the DiskExponential component.

tfamily \textcolor{red}{\textless type(stellarLuminosities)\textgreater} luminositiesStellarIsGettable() Get the luminositiesStellar property of the disk component.

tfamily \textcolor{red}{\textless logical\textgreater} luminositiesStellarIsSettable() Specify whether the luminositiesStellar property of the disk component is settable.

tfamily \textcolor{red}{\textless type(stellarLuminosities)\textgreater} luminositiesStellarRate(<type(stellarLuminosities)> value) Cumulate to the rate of the luminositiesStellar property of the DiskExponential component.
16. Coding GALACTICUS

tfamily \ void luminositiesStellarScale(<type(stellarLuminosities)> value) Set the scale of the luminositiesStellar property of the DiskExponential component.

tfamily \ void luminositiesStellarSet(<type(stellarLuminosities)> value) Set the luminositiesStellar property of the disk component.

tfamily \textcolor{red}{\textless double\textgreater} massGas() Get the massGas property of the disk component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)} massGasAttributeMatch(<logical>[requireGettable] → , <logical>[requireSettable] → , <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the massGas property of the disk component

tfamily \intzero massGasCount() Compute the count of evolvable quantities in the massGas property of the DiskExponential component.

tfamily \textcolor{red}{\textless double\textgreater} massGasGettable() Get the massGas property of the disk component.

tfamily \logicalzero massGasIsSettable() Specify whether the massGas property of the disk component is settable.

tfamily \ void massGasRate(<double> value) Cumulate to the rate of the massGas property of the DiskExponential component.

tfamily \ void massGasScale(<double> value) Set the scale of the massGas property of the DiskExponential component.

tfamily \ void massGasSet(<double> value) Set the massGas property of the disk component.

tfamily \textcolor{red}{\textless double\textgreater} massStellar() Get the massStellar property of the disk component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)} massStellarAttributeMatch(<logical>[requireGettable] → , <logical>[requireSettable] → , <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the massStellar property of the disk component

tfamily \intzero massStellarCount() Compute the count of evolvable quantities in the massStellar property of the DiskExponential component.

tfamily \textcolor{red}{\textless double\textgreater} massStellarGettable() Get the massStellar property of the disk component.

tfamily \logicalzero massStellarIsSettable() Specify whether the massStellar property of the disk component is settable.

tfamily \ void massStellarRate(<double> value) Cumulate to the rate of the massStellar property of the DiskExponential component.

tfamily \ void massStellarScale(<double> value) Set the scale of the massStellar property of the DiskExponential component.

tfamily \ void massStellarSet(<double> value) Set the massStellar property of the disk component.

tfamily \ void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.
logicalzero nullIsActive() Return whether the null implementation of the disk component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty, <integer> integerBufferCount, <integer> integerBuffer, <integer> doubleProperty, <integer> doubleBufferCount, <double> doubleBuffer, <double> time, <integer> instance) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount, <integer> doublePropertyCount, <double> time, <integer> instance) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty, <char> integerPropertyNames, <char> integerPropertyComments, <double> integerPropertyUnitsSI, <integer> doubleProperty, <char> doublePropertyNames, <char> doublePropertyComments, <double> doublePropertyUnitsSI, <double> time, <integer> instance) Generate names of outputtable properties.

doublezeropotential(<double> radius, <componentType> componentType, <massType> massType, <logical> haloLoaded) Compute the gravitational potential.

textcolor{red}{\textless double\textgreater} radius() Get the radius property of the disk component.

textcolor{red}{\textless varying\_string\(\rangle\)textgreater} radiusAttributeMatch(<logical> requireGettable, <logical> requireSettable, <logical> requireEvolvable) Return a list of implementations that provide the given list off attributes for the radius property of the disk component.

textcolor{red}{\textless double\textgreater} radiusIsGettable() Get the radius property of the disk component.

logicalzero radiusIsSettable() Specify whether the radius property of the disk component is settable.

tfamily \void radiusSet(<double> value) Set the radius property of the disk component.

doublezerorotationCurve(<double> radius, <componentType> componentType, <massType> massType, <logical> haloLoaded) Compute the rotation curve.

doublezerorotationCurveGradient(<double> radius, <componentType> componentType, <massType> massType, <logical> haloLoaded) Compute the rotation curve gradient.

intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double> array) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double> array) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double> array) Serialize the evolvable quantities to an array.

starFormationHistory() Get the starFormationHistory property of the disk component.
16. Coding GALACTICUS

tfamily \textcolor{red}{\textless textless type(varying \_string):\textgreater} starFormationHistoryAttributeM-
atch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Re-
turn a list of implementations that provide the given list off attributes for the starFormationHistory
property of the disk component.

tfamily \intzerostarFormationHistoryCount() Compute the count of evolvable quantities in the starFormationHistory
property of the DiskExponential component.

tfamily \textcolor{red}{\textless textless type(history):\textgreater} starFormationHistoryIsGettable() Get the starFormationHistory
property of the disk component.

tfamily \logicalzero starFormationHistoryIsSettable() Specify whether the starFormationHistory property
of the disk component is settable.

tfamily \void starFormationHistoryRate(<type(history)> value) Cumulate to the rate of the starFormationHistory
property of the DiskExponential component.

tfamily \void starFormationHistoryScale(<type(history)> value) Set the scale of the starFormationHistory
property of the DiskExponential component.

tfamily \void starFormationHistorySet(<type(history)> value) Set the starFormationHistory property
of the disk component.

tfamily \textcolor{red}{\textless double\textgreater} starFormationRate() Get the starFormationRate
property of the disk component.

tfamily \textcolor{red}{\textless textless type(varying \_string):\textgreater} starFormationRateAttributeM-
atch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Re-
turn a list of implementations that provide the given list off attributes for the starFormationRate
property of the disk component.

tfamily \intzerostarFormationRateCount() Compute the count of evolvable quantities in the starFormationRate
property of the DiskExponential component.

tfamily \textcolor{red}{\textless textless type(history):\textgreater} starFormationRateIsGettable() Get the starFormationRate
property of the disk component.

tfamily \logicalzero starFormationRateIsSettable() Specify whether the starFormationRate property
of the disk component is settable.

tfamily \void starFormationRateRate(<type(history)> value) Cumulate to the rate of the starFormationRate
property of the DiskExponential component.

tfamily \void starFormationRateScale(<type(history)> value) Set the scale of the starFormationRate
property of the DiskExponential component.

tfamily \void starFormationRateSet(<type(history)> value) Set the starFormationRate property
of the disk component.

tfamily \textcolor{red}{\textless type(history):\textgreater} stellarPropertiesHistory() Get the stellarPropertiesHistory
property of the disk component.

tfamily \textcolor{red}{\textless textless type(varying \_string):\textgreater} stellarPropertiesHistoryAttributeM-
atch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Re-
turn a list of implementations that provide the given list off attributes for the stellarPropertiesHistory
property of the disk component.

tfamily \intzerostellarPropertiesHistoryCount() Compute the count of evolvable quantities in the stellarPropertiesHistory
property of the DiskExponential component.

tfamily \textcolor{red}{\textless textless type(history):\textgreater} stellarPropertiesHistoryIsGettable() Get the stellarPropertiesHistory
property of the disk component.

tfamily \logicalzero stellarPropertiesHistoryIsSettable() Specify whether the stellarPropertiesHistory
property of the disk component is settable.

tfamily \void stellarPropertiesHistoryRate(<type(history)> value) Cumulate to the rate of the stellarPropertiesHistory
property of the DiskExponential component.
tfamily \void stellarPropertiesHistoryScale(<type(history)> value) Set the scale of the stellarPropertiesHistory property of the DiskExponential component.

tfamily \void stellarPropertiesHistorySet(<type(history)> value) Set the stellarPropertiesHistory property of the disk component.

tfamily \double zero surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

ntfamily \textcolor{red}{\textless double\textgreater} velocity() Get the velocity property of the disk component.

tfamily \textcolor{red}{\textless type(varying\_string):(\:)\textgreater} velocityAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the velocity property of the disk component.

tfamily \textcolor{red}{\textless double\textgreater} velocityIsGettable() Get the velocity property of the disk component.

tfamily \logical zero velocityIsSettable() Specify whether the velocity property of the disk component is settable.

tfamily \void velocitySet(<double> value) Set the velocity property of the disk component.

tfamily \logical zero verySimpleIsActive() Return whether the verySimple implementation of the disk component class is active.

ntfamily nodeComponentDiskExponential

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesGas() Get the abundancesGas property of the disk component.

tfamily \textcolor{red}{\textless type(abundances):(\:)\textgreater} abundancesGasAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the abundancesGas property of the disk component.

tfamily \int zero abundancesGasCount() Compute the count of evolvable quantities in the abundancesGas property of the DiskExponential component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesGasIsGettable() Get the abundancesGas property of the disk component.

tfamily \logical zero abundancesGasIsSettable() Specify whether the abundancesGas property of the disk component is settable.

tfamily \void abundancesGasRate(<type(abundances)> value) Cumulate to the rate of the abundancesGas property of the DiskExponential component.

tfamily \void abundancesGasScale(<type(abundances)> value) Set the scale of the abundancesGas property of the DiskExponential component.
16. Coding Galacticus

tfamily \void abundancesGasSet(<\textsf{type(abundances)}>) value Set the abundancesGas property of the disk component.

tfamily \textcolor{red}{\less than \textsf{type(abundances)}} \textgreater \ abundancesStellar() Get the abundancesStellar property of the disk component.

tfamily \textcolor{red}{\textless \textsf{type(varying\string)}(::)} \textgreater \ abundancesStellarAttributeMatch(<\textsf{logical}>[\textsf{requireGettable}]\rightarrow, <\textsf{logical}>[\textsf{requireSettable}]\rightarrow, <\textsf{logical}>[\textsf{requireEvolvable}]\rightarrow) Return a list of implementations that provide the given list of attributes for the abundancesStellar property of the disk component.

tfamily \intzero abundancesStellarCount() Compute the count of evolvable quantities in the abundancesStellar property of the DiskExponential component.

tfamily \textcolor{red}{\textless \textsf{double}} \textgreater \ abundancesStellarIsGettable() Get the abundancesStellar property of the disk component.

tfamily \logicalzero abundancesStellarIsSettable() Specify whether the abundancesStellar property of the disk component is settable.

tfamily \void abundancesStellarRate(<\textsf{type(abundances)}>) value Cumulate to the rate of the abundancesStellar property of the DiskExponential component.

tfamily \void abundancesStellarScale(<\textsf{type(abundances)}>) value Set the scale of the abundancesStellar property of the DiskExponential component.

tfamily \void abundancesStellarSet(<\textsf{type(abundances)}>) value Set the abundancesStellar property of the disk component.

tfamily \textcolor{red}{\textless \textsf{double}} \textgreater \ angularMomentum() Get the angularMomentum property of the disk component.

tfamily \textcolor{red}{\textless \textsf{type(varying\string)}(::)} \textgreater \ angularMomentumAttributeMatch(<\textsf{logical}>[\textsf{requireGettable}]\rightarrow, <\textsf{logical}>[\textsf{requireSettable}]\rightarrow, <\textsf{logical}>[\textsf{requireEvolvable}]\rightarrow) Return a list of implementations that provide the given list of attributes for the angularMomentum property of the disk component.

tfamily \intzero angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the DiskExponential component.

tfamily \textcolor{red}{\textless \textsf{double}} \textgreater \ angularMomentumIsGettable() Get the angularMomentum property of the disk component.

tfamily \logicalzero angularMomentumIsSettable() Specify whether the angularMomentum property of the disk component is settable.

tfamily \void angularMomentumRate(<\textsf{double}>) value Cumulate to the rate of the angularMomentum property of the DiskExponential component.

tfamily \void angularMomentumScale(<\textsf{double}>) value Set the scale of the angularMomentum property of the DiskExponential component.

tfamily \void angularMomentumSet(<\textsf{double}>) value Set the angularMomentum property of the disk component.

tfamily \void attachPipes() Attach pipes to the exponential disk component.
16.5. Objects

tfamily void builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily double zero density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

tfamily void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily void destroy() Destroy the object.

tfamily void dump() Generate an ASCII dump of all properties.

tfamily void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily void dumpXML() Generate an XML dump of all properties.

tfamily double zero enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily logical zero exponentialIsActive() Return whether the exponential implementation of the disk component class is active.

tfamily textcolor{red}{	extless double\textgreater} halfMassRadius() Get the halfMassRadius property of the disk component.

tfamily textcolor{red}{	extless type(varying\_string)(:)}\textgreater halfMassRadiusAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the halfMassRadius property of the disk component.

tfamily textcolor{red}{	extless double\textgreater} halfMassRadiusIsGettable() Get the halfMassRadius property of the disk component.

tfamily logical zero halfMassRadiusIsSettable() Specify whether the halfMassRadius property of the disk component is settable.

tfamily textcolor{red}{	extless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily void initialize() Initialize the object.

tfamily textcolor{red}{	extless logical\textgreater} isInitialized() Get the isInitialized property of the disk component.
16. Coding GALACTICUS

tfamily {textcolor{red}{\textless textless type(varying \_ string)(::):textgreater}} isInitializedAttributeMatch(<logical>|requireGettable→, <logical>|requireSettable→, <logical>|requireEvolvable→) Return a list of implementations that provide the given list of attributes for the isInitialized property of the disk component.

tfamily isInitializedIsGettable() Get the isInitialized property of the disk component.

tfamily {logicalzero} isInitializedIsSettable() Specify whether the isInitialized property of the disk component is settable.

tfamily {void} isInitializedSet(<logical> value) Set the isInitialized property of the disk component.

tfamily {textless type(stellarLuminosities):textgreater} luminositiesStellar() Get the luminositiesStellar property of the disk component.

tfamily luminositiesStellarAttributeMatch(<logical>|requireGettable→, <logical>|requireSettable→, <logical>|requireEvolvable→) Return a list of implementations that provide the given list of attributes for the luminositiesStellar property of the disk component.

tfamily {intzero} luminositiesStellarCount() Compute the count of evolvable quantities in the luminositiesStellar property of the DiskExponential component.

tfamily {textless type(stellarLuminosities):textgreater} luminositiesStellarIsGettable() Get the luminositiesStellar property of the disk component.

tfamily {logicalzero} luminositiesStellarIsSettable() Specify whether the luminositiesStellar property of the disk component is settable.

tfamily {void} luminositiesStellarRate(<type(stellarLuminosities)> value) Cumulate to the rate of the luminositiesStellar property of the DiskExponential component.

tfamily {void} luminositiesStellarScale(<type(stellarLuminosities)> value) Set the scale of the luminositiesStellar property of the DiskExponential component.

tfamily {void} luminositiesStellarSet(<type(stellarLuminosities)> value) Set the luminositiesStellar property of the disk component.

tfamily {textcolor{red}{\textless double:textgreater}} massGas() Get the massGas property of the disk component.

tfamily massGasAttributeMatch(<logical>|requireGettable→, <logical>|requireSettable→, <logical>|requireEvolvable→) Return a list of implementations that provide the given list of attributes for the massGas property of the disk component.

tfamily {intzero} massGasCount() Compute the count of evolvable quantities in the massGas property of the DiskExponential component.

tfamily {textless double:textgreater} massGasIsGettable() Get the massGas property of the disk component.

tfamily {logicalzero} massGasIsSettable() Specify whether the massGas property of the disk component is settable.

tfamily {void} massGasRate(<double> value) Cumulate to the rate of the massGas property of the DiskExponential component.
tfamily
  \textcolor{red}{\textgreater \textless \textgreater} massGasScale(<\text{double}>) Set the scale of the massGas property of the DiskExponential component.

tfamily
  \textcolor{red}{\textgreater \textless \textgreater} massGasSet(<\text{double}>) Set the massGas property of the disk component.

tfamily
  \textcolor{red}{\textless \textgreater} massStellar() Get the massStellar property of the disk component.

tfamily
  \textcolor{red}{\textless \textgreater \textless \textgreater} massStellarAttributeMatch(<\text{logical}>)\texttt{[requireGettable]} →, <\text{logical}>)\texttt{[requireSettable]} →, <\text{logical}>)\texttt{[requireEvolvable]} →) Return a list of implementations that provide the given list off attributes for the massStellar property of the disk component.

tfamily
  \textcolor{red}{\textless \textgreater \textless \textgreater} massStellarCount() Compute the count of evolvable quantities in the massStellar property of the DiskExponential component.

tfamily
  \textcolor{red}{\textless \textgreater \textless \textgreater} massStellarIsGettable() Get the massStellar property of the disk component.

tfamily
  \logical\textcolor{red}{\textless \textgreater \textless \textgreater} massStellarIsSettable() Specify whether the massStellar property of the disk component is settable.

tfamily
  \textcolor{red}{\textless \textgreater \textless \textgreater} massStellarRate(<\text{double}>) Cumulate to the rate of the massStellar property of the DiskExponential component.

tfamily
  \textcolor{red}{\textless \textgreater \textless \textgreater} massStellarScale(<\text{double}>) Set the scale of the massStellar property of the DiskExponential component.

tfamily
  \textcolor{red}{\textless \textgreater \textless \textgreater} massStellarSet(<\text{double}>) Set the massStellar property of the disk component.

tfamily
  \textcolor{red}{\textless \textgreater \textless \textgreater} nameFromIndex(<\text{integer}>)\texttt{[count]} →, <\text{varying\_string}>\texttt{name}←) Return the name of a property given is index.

tfamily
  \logical\textcolor{red}{\textless \textgreater \textless \textgreater} nullIsActive() Return whether the null implementation of the disk component class is active.

tfamily
  \textcolor{red}{\textless \textgreater \textless \textgreater} odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily
  \textcolor{red}{\textless \textgreater \textless \textgreater} odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily
  \textcolor{red}{\textless \textgreater \textless \textgreater} output(<\text{integer}>)\texttt{[integerProperty]} →, <\text{integer}>\texttt{[integerBufferCount]}→, <\text{integer}>)\texttt{[integerBuffer]} ↔, <\text{integer}>)\texttt{[doubleProperty]} →, <\text{integer}>\texttt{[doubleBufferCount]}→, <\text{double}>)\texttt{[doubleBuffer]} ↔, <\text{integer}>)\texttt{[time]} →, <\text{integer}>)\texttt{[instance]}→) Generate values of outputtable properties.

tfamily
  \textcolor{red}{\textless \textgreater \textless \textgreater} outputCount(<\text{integer}>)\texttt{[integerPropertyCount]}→, <\text{integer}>\texttt{[doublePropertyCount]}→, <\text{double}>)\texttt{[time]} →, <\text{integer}>)\texttt{[instance]}→) Compute a count of outputtable properties.

tfamily
  \textcolor{red}{\textless \textgreater \textless \textgreater} outputNames(<\text{integer}>)\texttt{[integerProperty]}→, <\text{char\[*\]}>\texttt{[integerPropertyNames]}→, <\text{char\[*\]]}>\texttt{[integerPropertyComments]}→, <\text{char\[*\]]}>\texttt{[integerPropertyUnits\_SI]}→, <\text{integer}>)\texttt{[doubleProperty]}→, <\text{char\[*\]}>\texttt{[doublePropertyNames]}→, <\text{char\[*\]]}>\texttt{[doublePropertyComments]}→, <\text{char\[*\]]}>\texttt{[doublePropertyUnits\_SI]}→, <\text{double}>)\texttt{[time]} →, <\text{integer}>)\texttt{[instance]}→) Generate names of outputtable properties.

tfamily
  \textcolor{red}{\textless \textgreater \textless \textgreater} potential(<\text{double}>)\texttt{[radius]} →, <\text{componentType}>\texttt{[componentType]} →, <\text{massType}>\texttt{[massType]} →, <\text{logical}>)\texttt{[halo\_loaded]} →) Compute the gravitational potential.
tfamily \textcolor{red}\{	extless double\textgreater\} radius() Get the \texttt{radius} property of the \texttt{disk} component.

tfamily \textcolor{red}\{\textless double\textgreater\} radiusAttributeMatch(<logical>[requireGettable]\rightarrow, <logical>[requireSettable]\rightarrow, <logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the \texttt{radius} property of the \texttt{disk} component.

tfamily \textcolor{red}\{\textless double\textgreater\} radiusIsGettable() Get the \texttt{radius} property of the \texttt{disk} component.

tfamily \textcolor{red}\{\textless double\textgreater\} radiusIsSettable() Specify whether the \texttt{radius} property of the \texttt{disk} component is settable.

tfamily \void radiusSet(<double> value) Set the \texttt{radius} property of the \texttt{disk} component.

tfamily \void readRaw(<integer> fileHandle\rightarrow) Read a binary dump of the \texttt{nodeComponent} from the given \texttt{fileHandle}.

tfamily \double zero rotationCurve(<double> radius\rightarrow, <componentType>[componentType]\rightarrow, <massType>[massType]\rightarrow, <logical>[haloLoaded]\rightarrow) Compute the rotation curve.

tfamily \double zero rotationCurveGradient(<double> radius\rightarrow, <componentType>[componentType]\rightarrow, <massType>[massType]\rightarrow, <logical>[haloLoaded]\rightarrow) Compute the rotation curve gradient.

tfamily \int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array\rightarrow) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array\rightarrow) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array\rightarrow) Serialize the evolvable quantities to an array.

tfamily \textcolor{red}\{\textless type\textgreater\} starFormationHistory() Get the \texttt{starFormationHistory} property of the \texttt{disk} component.

tfamily \textcolor{red}\{\textless type\textgreater\} starFormationHistoryAttributeMatch(<logical>[requireGettable]\rightarrow, <logical>[requireSettable]\rightarrow, <logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the \texttt{starFormationHistory} property of the \texttt{disk} component.

tfamily \int zero starFormationHistoryCount() Compute the count of evolvable quantities in the \texttt{starFormationHistory} property of the \texttt{DiskExponential} component.

tfamily \textcolor{red}\{\textless type\textgreater\} starFormationHistoryIsGettable() Get the \texttt{starFormationHistory} property of the \texttt{disk} component.

tfamily \logical zero starFormationHistoryIsSettable() Specify whether the \texttt{starFormationHistory} property of the \texttt{disk} component is settable.

tfamily \void starFormationHistoryRate(<type\textgreater\text{history}> value) Cumulate to the rate of the \texttt{starFormationHistory} property of the \texttt{DiskExponential} component.

tfamily \void starFormationHistoryScale(<type\textgreater\text{history}> value) Set the scale of the \texttt{starFormationHistory} property of the \texttt{DiskExponential} component.

tfamily \void starFormationHistorySet(<type\textgreater\text{history}> value) Set the \texttt{starFormationHistory} property of the \texttt{disk} component.
16.5. Objects

tfamily \textcolor{red}{\textless double\textgreater} starFormationRate() Get the starFormationRate property of the disk component.

tfamily \textcolor{red}{\textless type(varying\_string):\textgreater} starFormationRateAttributeMatch(<logical>[requireGettable]\rightarrow, <logical>[requireSettable]\rightarrow, <logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the starFormationRate property of the disk component.

tfamily \void starFormationRateFunction(<function()> deferredFunction) Set the function to be used for the get method of the starFormationRate property of the DiskExponential component.

tfamily \logicalzero starFormationRateIsAttached() Return whether the get method of the starFormationRate property of the DiskExponential component has been attached to a function.

tfamily \textcolor{red}{\textless double\textgreater} starFormationRateIsGettable() Get the starFormationRate property of the disk component.

tfamily \logicalzero starFormationRateIsSettable() Specify whether the starFormationRate property of the disk component is settable.

tfamily \textcolor{red}{\textless type(history):\textgreater} stellarPropertiesHistory() Get the stellarPropertiesHistory property of the disk component.

tfamily \textcolor{red}{\textless type(varying\_string):\textgreater} stellarPropertiesHistoryAttributeMatch(<logical>[requireGettable]\rightarrow, <logical>[requireSettable]\rightarrow, <logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the stellarPropertiesHistory property of the disk component.

tfamily \intzero stellarPropertiesHistoryCount() Compute the count of evolvable quantities in the stellarPropertiesHistory property of the DiskExponential component.

tfamily \textcolor{red}{\textless type(history):\textgreater} stellarPropertiesHistoryIsGettable() Get the stellarPropertiesHistory property of the disk component.

tfamily \logicalzero stellarPropertiesHistoryIsSettable() Specify whether the stellarPropertiesHistory property of the disk component is settable.

tfamily \void stellarPropertiesHistoryRate(<type(history)> value) Cumulate to the rate of the stellarPropertiesHistory property of the DiskExponential component.

tfamily \void stellarPropertiesHistoryScale(<type(history)> value) Set the scale of the stellarPropertiesHistory property of the DiskExponential component.

tfamily \void stellarPropertiesHistorySet(<type(history)> value) Set the stellarPropertiesHistory property of the disk component.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical\rightarrow, <componentType>[componentType]\rightarrow, <massType>[massType]\rightarrow, <weightBy>[weightBy]\rightarrow, <integer>[weightIndex]\rightarrow, <logical>[haloLoaded]\rightarrow) Compute the surface density.

tfamily \textcolor{red}{\textless type(varying\_string):\textgreater} type() Return the type of this object.

tfamily \textcolor{red}{\textless double\textgreater} velocity() Get the velocity property of the disk component.
16. Coding Galacticus

tfamily \textcolor{red}{\textless type(varying \_string)(:)\textgreater} velocityAttributeMatch(<\text{logical}>[requireGettable]→, <\text{logical}>[requireSettable]→, <\text{logical}>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the velocity property of the disk component.

tfamily \textcolor{red}{\textless double\textgreater} velocityIsGettable() Get the velocity property of the disk component.

tfamily \textcolor{red}{\textless logical\textgreater} velocityIsSettable() Specify whether the velocity property of the disk component is settable.

tfamily \text{void} velocitySet(<double> value) Set the velocity property of the disk component.

tfamily \textcolor{red}{\textless logical\textgreater} verySimpleIsActive() Return whether the verySimple implementation of the disk component class is active.

tfamily nodeComponentDiskNull

tfamily \textcolor{red}{\textless type(\text{abundances})\textgreater} abundancesGas() Get the abundancesGas property of the disk component.

tfamily \textcolor{red}{\textless type(varying \_string)(:)\textgreater} abundancesGasAttributeMatch(<\text{logical}>[requireGettable]→, <\text{logical}>[requireSettable]→, <\text{logical}>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the abundancesGas property of the disk component.

tfamily \text{int} zero abundancesGasCount() Compute the count of evolvable quantities in the abundancesGas property of the DiskExponential component.

tfamily \textcolor{red}{\textless type(\text{abundances})\textgreater} abundancesGasIsGettable() Get the abundancesGas property of the disk component.

tfamily \textcolor{red}{\textless logical\textgreater} abundancesGasIsSettable() Specify whether the abundancesGas property of the disk component is settable.

tfamily \text{void} abundancesGasRate(<\text{type(\text{abundances})}> value) Cumulate to the rate of the abundancesGas property of the DiskExponential component.

tfamily \text{void} abundancesGasScale(<\text{type(\text{abundances})}> value) Set the scale of the abundancesGas property of the DiskExponential component.

tfamily \text{void} abundancesGasSet(<\text{type(\text{abundances})}> value) Set the abundancesGas property of the disk component.

tfamily \textcolor{red}{\textless type(\text{abundances})\textgreater} abundancesStellar() Get the abundancesStellar property of the disk component.

tfamily \textcolor{red}{\textless type(varying \_string)(:)\textgreater} abundancesStellarAttributeMatch(<\text{logical}>[requireGettable]→, <\text{logical}>[requireSettable]→, <\text{logical}>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the abundancesStellar property of the disk component.

tfamily \text{int} zero abundancesStellarCount() Compute the count of evolvable quantities in the abundancesStellar property of the DiskExponential component.

tfamily \textcolor{red}{\textless type(\text{abundances})\textgreater} abundancesStellarIsGettable() Get the abundancesStellar property of the disk component.
16.5. Objects

tfamily \logicalzero abundancesStellarIsSettable() Specify whether the abundancesStellar property of the disk component is settable.

tfamily \void abundancesStellarRate(<type(abundances)> value) Cumulate to the rate of the abundancesStellar property of the DiskExponential component.

tfamily \void abundancesStellarScale(<type(abundances)> value) Set the scale of the abundancesStellar property of the DiskExponential component.

tfamily \void abundancesStellarSet(<type(abundances)> value) Set the abundancesStellar property of the disk component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentum() Get the angularMomentum property of the disk component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater angularMomentumAttributeMatch(<logical>[requireGettable→, <logical>[requireSettable→, <logical>[requireEvolvable→) Return a list of implementations that provide the given list of attributes for the angularMomentum property of the disk component.

tfamily \intzero angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the DiskExponential component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumIsGettable() Get the angularMomentum property of the disk component.

tfamily \logicalzero angularMomentumIsSettable() Specify whether the angularMomentum property of the disk component is settable.

tfamily \void angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the DiskExponential component.

tfamily \void angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the DiskExponential component.

tfamily \void angularMomentumSet(<double> value) Set the angularMomentum property of the disk component.

tfamily \void builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily \doublezero density(<double(3)> positionSpherical→, <componentType>[componentType→, <massType>[massType→, <weightBy>[weightBy→, <integer>[weightIndex→, <logical>[haloLoaded→) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.
16. Coding GALACTICUS

tfamily void dumpXML() Generate an XML dump of all properties.

tfamily double zero enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily logical zero exponentialIsActive() Return whether the exponential implementation of the disk component class is active.

tfamily textcolor{red}{	extless double\textgreater} halfMassRadius() Get the halfMassRadius property of the disk component.

tfamily textcolor{red}{	extless logical\textgreater} halfMassRadiusIsGettable() Get the halfMassRadius property of the disk component.

tfamily logical zero halfMassRadiusIsSettable() Specify whether the halfMassRadius property of the disk component is settable.

tfamily textcolor{red}{	extless *type(treeNode)\textgreater} host() Return a pointer to the host TreeNode object.

tfamily void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily void initialize() Initialize the object.

tfamily textcolor{red}{	extless logical\textgreater} isInitialized() Get the isInitialized property of the disk component.

tfamily textcolor{red}{	extless logical\textgreater} isInitializedIsGettable() Get the isInitialized property of the disk component.

tfamily logical zero isInitializedIsSettable() Specify whether the isInitialized property of the disk component is settable.

tfamily void isInitializedSet(<logical> value) Set the isInitialized property of the disk component.

tfamily textcolor{red}{	extless type(stellarLuminosities)\textgreater} luminositiesStellar() Get the luminositiesStellar property of the disk component.
16.5. Objects

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater luminositiesStellarAttributeMatch(<logical>[requireGettable] → , <logical>[requireSettable] → , <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the luminositiesStellar property of the disk component.

tfamily \textcolor{red}{\textless double\textgreater} massGas() Get the massGas property of the disk component.

tfamily \textcolor{red}{\textless double\textgreater} massStellar() Get the massStellar property of the disk component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)}\textgreater massStellarAttributeMatch(<logical>[requireGettable] → , <logical>[requireSettable] → , <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the massStellar property of the disk component.

tfamily \intzero luminositiesStellarCount() Compute the count of evolvable quantities in the luminositiesStellar property of the DiskExponential component.

tfamily \textcolor{red}{\textless type(stellarLuminosities)\textgreater} luminositiesStellarIsGettable() Get the luminositiesStellar property of the disk component.

tfamily \textcolor{red}{\textless double\textgreater} massGasIsGettable() Get the massGas property of the disk component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)}\textgreater massGasAttributeMatch(<logical>[requireGettable] → , <logical>[requireSettable] → , <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the massGas property of the disk component.

tfamily \intzero massGasCount() Compute the count of evolvable quantities in the massGas property of the DiskExponential component.

tfamily \textcolor{red}{\textless type(stellarLuminosities)\textgreater} luminositiesStellarIsSettable() Specify whether the luminositiesStellar property of the disk component is settable.

tfamily \textcolor{red}{\textless double\textgreater} massStellarIsGettable() Get the massStellar property of the disk component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)}\textgreater massStellarAttributeMatch(<logical>[requireGettable] → , <logical>[requireSettable] → , <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the massStellar property of the disk component.

tfamily \intzero massStellarCount() Compute the count of evolvable quantities in the massStellar property of the DiskExponential component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)}\textgreater massStellarAttributeMatch(<logical>[requireGettable] → , <logical>[requireSettable] → , <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the massStellar property of the disk component.

tfamily \intzero massStellarCount() Compute the count of evolvable quantities in the massStellar property of the DiskExponential component.
tfamily \textcolor{red}{\textless\textgreater} massStellarIsGettable() Get the \texttt{massStellar} property of the \texttt{disk} component.

tfamily \textcolor{red}{\textless\textgreater} massStellarIsSettable() Specify whether the \texttt{massStellar} property of the \texttt{disk} component is settable.

tfamily \textcolor{red}{\textless\textgreater} massStellarRate(<double> value) Cumulate to the rate of the \texttt{massStellar} property of the \texttt{DiskExponential} component.

tfamily \textcolor{red}{\textless\textgreater} massStellarScale(<double> value) Set the scale of the \texttt{massStellar} property of the \texttt{DiskExponential} component.

tfamily \textcolor{red}{\textless\textgreater} massStellarSet(<double> value) Set the \texttt{massStellar} property of the \texttt{disk} component.

tfamily nullIsActive() Return whether the null implementation of the disk component class is active.

tfamily odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily output(<integer> integerProperty\leftrightarrow, <integer> integerBufferCount\leftrightarrow, <integer[:,:] > integerBuffer\leftrightarrow, <integer> doubleProperty\leftrightarrow, <integer> doubleBufferCount\leftrightarrow, <double[:,:] > doubleBuffer\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow) Generate values of outputtable properties.

tfamily outputCount(<integer> integerPropertyCount\leftrightarrow, <integer> doublePropertyCount\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow) Compute a count of outputtable properties.

tfamily outputNames(<integer> integerProperty\leftrightarrow, <char[\*](:) > integerPropertyNames\leftrightarrow, <char[\*](:) > integerPropertyComments\leftrightarrow, <double(:) > integerPropertyUnitsSI\leftrightarrow, <integer> doubleProperty\leftrightarrow, <char[\*](:) > doublePropertyNames\leftrightarrow, <char[\*](:) > doublePropertyComments\leftrightarrow, <double(:) > doublePropertyUnitsSI\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow) Generate names of outputtable properties.

double zero potential(<double> radius\rightarrow, <componentType> [componentType]\rightarrow, <massType> [massType]\rightarrow, <logical> [haloLoaded]\rightarrow) Compute the gravitational potential.

tfamily \textcolor{red}{\textless\textgreater} radius() Get the \texttt{radius} property of the \texttt{disk} component.

tfamily \textcolor{red}{\textless\textgreater} radiusAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list off attributes for the \texttt{radius} property of the \texttt{disk} component.

tfamily \textcolor{red}{\textless\textgreater} radiusIsGettable() Get the \texttt{radius} property of the \texttt{disk} component.

tfamily \textcolor{red}{\textless\textgreater} radiusIsSettable() Specify whether the \texttt{radius} property of the \texttt{disk} component is settable.

tfamily void radiusSet(<double> value) Set the \texttt{radius} property of the \texttt{disk} component.
16.5. Objects

tfamily \void readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezero rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

tfamily \textcolor{red}{\textlesstype(history)\textgreater} starFormationHistory() Get the starFormationHistory property of the disk component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)\textgreater} starFormationHistoryAttributeMatch(<logical> [requireGettable]→, <logical> [requireSettable]→, <logical> [requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the starFormationHistory property of the disk component.

tfamily \intzero starFormationHistoryCount() Compute the count of evolvable quantities in the starFormationHistory property of the DiskExponential component.

tfamily \textcolor{red}{\textless \texttype(history)\textgreater} starFormationHistoryIsGettable() Get the starFormationHistory property of the disk component.

tfamily \logicalzero starFormationHistoryIsSettable() Specify whether the starFormationHistory property of the disk component is settable.

tfamily \void starFormationHistoryRate(<type(history)> value) Cumulate to the rate of the starFormationHistory property of the DiskExponential component.

tfamily \void starFormationHistoryScale(<type(history)> value) Set the scale of the starFormationHistory property of the DiskExponential component.

tfamily \void starFormationHistorySet(<type(history)> value) Set the starFormationHistory property of the disk component.

tfamily \textcolor{red}{\textless \textdouble\textgreater} starFormationRate() Get the starFormationRate property of the disk component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:\textgreater)} starFormationRateAttributeMatch(<logical> [requireGettable]→, <logical> [requireSettable]→, <logical> [requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the starFormationRate property of the disk component.

tfamily \textcolor{red}{\textless \textdouble\textgreater} starFormationRateIsGettable() Get the starFormationRate property of the disk component.

tfamily \logicalzero starFormationRateIsSettable() Specify whether the starFormationRate property of the disk component is settable.
16. Coding GALACTICUS

tfamily {textcolor{red}}\{\textless type(history)\textgreater\} stellarPropertiesHistory() Get the stellarPropertiesHistory property of the disk component.

tfamily {textcolor{red}}\{\textless varying\_string\} stellarPropertiesHistoryAttributeMatch(\textless logical\textgreater [requireGettable] →, \textless logical\textgreater [requireSettable] →, \textless logical\textgreater [requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the stellarPropertiesHistory property of the disk component.

tfamily \intzero stellarPropertiesHistoryCount() Compute the count of evolvable quantities in the stellarPropertiesHistory property of the DiskExponential component.

tfamily {textcolor{red}}\{\textless type(history)\textgreater\} stellarPropertiesHistoryIsGettable() Get the stellarPropertiesHistory property of the disk component.

tfamily \logicalzero stellarPropertiesHistoryIsSettable() Specify whether the stellarPropertiesHistory property of the disk component is settable.

tfamily \void stellarPropertiesHistoryRate(\textless type(history)\textgreater value) Cumulate to the rate of the stellarPropertiesHistory property of the DiskExponential component.

tfamily \void stellarPropertiesHistoryScale(\textless type(history)\textgreater value) Set the scale of the stellarPropertiesHistory property of the DiskExponential component.

tfamily \void stellarPropertiesHistorySet(\textless type(history)\textgreater value) Set the stellarPropertiesHistory property of the disk component.

tfamily {doublezero} surfaceDensity(\textless double(3)\textgreater positionCylindrical→, \textless componentType\textgreater [componentType]→, \textless massType\textgreater [massType]→, \textless weightBy\textgreater [weightBy]→, \textless integer\textgreater [weightIndex]→, \textless logical\textgreater [haloLoaded]→) Compute the surface density.

tfamily {textcolor{red}}\{\textless varying\_string\} type() Return the type of this object.

tfamily {textcolor{red}}\{\textless double\} velocity() Get the velocity property of the disk component.

tfamily {textcolor{red}}\{\textless varying\_string\} velocityAttributeMatch(\textless logical\textgreater [requireGettable] →, \textless logical\textgreater [requireSettable] →, \textless logical\textgreater [requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the velocity property of the disk component.

tfamily \logicalzero velocityIsSettable() Specify whether the velocity property of the disk component is settable.

tfamily \void velocitySet(\textless double\textgreater value) Set the velocity property of the disk component.

tfamily \logicalzero verySimpleIsActive() Return whether the verySimple implementation of the disk component class is active.
16.5. Objects

tfamily nodeComponentDiskVerySimple

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesGas() Get the abundancesGas property of the disk component.

tfamily \textcolor{red}{\textless type(varying\_string)()\textgreater} abundancesGasAttributeMatch (<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the abundancesGas property of the disk component.

tfamily \intzero abundancesGasCount() Compute the count of evolvable quantities in the abundancesGas property of the DiskExponential component.

tfamily \textcolor{red}{\textlesstype(abundances)\textgreater} abundancesGasIsGettable() Get the abundancesGas property of the disk component.

tfamily \logicalzero abundancesGasIsSettable() Specify whether the abundancesGas property of the disk component is settable.

tfamily \void abundancesGasRate(<type(abundances)> value) Cumulate to the rate of the abundancesGas property of the DiskExponential component.

tfamily \void abundancesGasScale(<type(abundances)> value) Set the scale of the abundancesGas property of the DiskExponential component.

tfamily \void abundancesGasSet(<type(abundances)> value) Set the abundancesGas property of the disk component.

tfamily \textcolor{red}{\textlesstype(abundances)\textgreater} abundancesStellar() Get the abundancesStellar property of the disk component.

tfamily \textcolor{red}{\textless type(varying\_string)()\textgreater} abundancesStellarAttributeMatch (<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the abundancesStellar property of the disk component.

tfamily \intzero abundancesStellarCount() Compute the count of evolvable quantities in the abundancesStellar property of the DiskExponential component.

tfamily \textcolor{red}{\textlesstype(abundances)\textgreater} abundancesStellarIsGettable() Get the abundancesStellar property of the disk component.

tfamily \logicalzero abundancesStellarIsSettable() Specify whether the abundancesStellar property of the disk component is settable.

tfamily \void abundancesStellarRate(<type(abundances)> value) Cumulate to the rate of the abundancesStellar property of the DiskExponential component.

tfamily \void abundancesStellarScale(<type(abundances)> value) Set the scale of the abundancesStellar property of the DiskExponential component.

tfamily \void abundancesStellarSet(<type(abundances)> value) Set the abundancesStellar property of the disk component.

tfamily \textcolor{red}{\textlesstype(double)\textgreater} angularMomentum() Get the angularMomentum property of the disk component.
Coding GALACTICUS

16. Coding GALACTICUS

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumIsGettable() Get the angularMomentum property of the disk component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumIsSettable() Specify whether the angularMomentum property of the disk component is settable.

tfamily \void angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the DiskExponential component.

tfamily \void angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the DiskExponential component.

tfamily \void angularMomentumSet(<double> value) Set the angularMomentum property of the disk component.

tfamily \void attachPipe() Attach pipes to the very simple disk component.

tfamily \void builder(<\text{node}>componentDefinition\rightarrow) Build a nodeComponent from a supplied XML definition.

tfamily \double zero density(<double (3)> positionSpherical\rightarrow, <\text{componentType}>[\text{componentType}]\rightarrow, <\text{massType}>[\text{massType}]\rightarrow, <\text{weightBy}>[\text{weightBy}]\rightarrow, <\text{integer}>[\text{weightIndex}]\rightarrow, <\text{logical}>[\text{haloLoaded}]\rightarrow) Compute the density.

tfamily \void deserializeRates(<double(:)> array\leftarrow) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array\leftarrow) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array\leftarrow) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<\text{integer}> fileHandle\rightarrow) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \double zero enclosedMass(<double> radius\rightarrow, <\text{componentType}>[\text{componentType}]\rightarrow, <\text{massType}>[\text{massType}]\rightarrow, <\text{weightBy}>[\text{weightBy}]\rightarrow, <\text{integer}>[\text{weightIndex}]\rightarrow, <\text{logical}>[\text{haloLoaded}]\rightarrow) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless logical\textgreater} exponentialIsActive() Return whether the exponential implementation of the disk component class is active.

tfamily \textcolor{red}{\textless double\textgreater} halfMassRadius() Get the halfMassRadius property of the disk component.
16.5. Objects

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} halfMassRadiusAttributeMatch(<logical>[requireGettable→, <logical>[requireSettable→, <logical>[requireEvolvable→]) Return a list of implementations that provide the given list off attributes for the \texttt{halfMassRadius} property of the disk component.

tfamily \textcolor{red}{\textless double\textgreater} halfMassRadiusIsGettable() Get the \texttt{halfMassRadius} property of the disk component.

tfamily \logicalzero halfMassRadiusIsSettable() Specify whether the \texttt{halfMassRadius} property of the disk component is settable.

tfamily \textcolor{red}{\textless type(treeNode)\textgreater} host() Return a pointer to the host \texttt{treeNode} object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the \texttt{hotHaloCoolingAngularMomentum} property of the \texttt{hotHalo} component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the \texttt{hotHaloCoolingMass} property of the \texttt{hotHalo} component.

tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}{\textless logical\textgreater} isInitialized() Get the \texttt{isInitialized} property of the disk component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} isInitializedAttributeMatch(<logical>[requireGettable→, <logical>[requireSettable→, <logical>[requireEvolvable→]) Return a list of implementations that provide the given list off attributes for the \texttt{isInitialized} property of the disk component.

tfamily \textcolor{red}{\textless logical\textgreater} isInitializedIsGettable() Get the \texttt{isInitialized} property of the disk component.

tfamily \logicalzero isInitializedIsSettable() Specify whether the \texttt{isInitialized} property of the disk component is settable.

tfamily \void isInitializedSet(<logical> value) Set the \texttt{isInitialized} property of the disk component.

tfamily \textcolor{red}{\textless type(stellarLuminosities)\textgreater} luminositiesStellar() Get the \texttt{luminositiesStellar} property of the disk component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} luminositiesStellarAttributeMatch(<logical>[requireGettable→, <logical>[requireSettable→, <logical>[requireEvolvable→]) Return a list of implementations that provide the given list off attributes for the \texttt{luminositiesStellar} property of the disk component.

tfamily \intzero luminositiesStellarCount() Compute the count of evolvable quantities in the \texttt{luminositiesStellar} property of the \texttt{DiskExponential} component.

tfamily \textcolor{red}{\textless type(stellarLuminosities)\textgreater} luminositiesStellarIsGettable() Get the \texttt{luminositiesStellar} property of the disk component.

tfamily \logicalzero luminositiesStellarIsSettable() Specify whether the \texttt{luminositiesStellar} property of the disk component is settable.
tfamily \void luminositiesStellarRate(<type(stellarLuminosities)> value) Cumulate to the rate of the luminositiesStellar property of the DiskExponential component.

tfamily \void luminositiesStellarScale(<type(stellarLuminosities)> value) Set the scale of the luminositiesStellar property of the DiskExponential component.

tfamily \void luminositiesStellarSet(<type(stellarLuminosities)> value) Set the luminositiesStellar property of the disk component.

tfamily \textcolor{red}{\textless double\textgreater} massGas() Get the massGas property of the disk component.

tfamily \textcolor{red}{\textless double\textgreater} massStellar() Get the massStellar property of the disk component.

tfamily \logicalzero massGasIsSettable() Specify whether the massGas property of the disk component is settable.

tfamily \void massGasRate(<double> value) Cumulate to the rate of the massGas property of the DiskExponential component.

tfamily \void massGasScale(<double> value) Set the scale of the massGas property of the DiskExponential component.

tfamily \void massStellarSet(<double> value) Set the massStellar property of the disk component.
tfamily \void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given its index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the disk component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount↔, <integer>doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty↔, <char[*](:)>integerPropertyNames↔, <char[*](:)>integerPropertyComments↔, <double(:)> integerPropertyUnitsSI↔, <integer>doubleProperty↔, <char[*](:)> doublePropertyNames↔, <char[*](:)>doublePropertyComments↔, <double(:)> doublePropertyUnitsSI↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \doublezeropotential(<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the gravitational potential.

tfamily \textcolor{red}{\textless double\textgreater} radius() Get the radius property of the disk component.

tfamily \textcolor{red}{\textless varying\_string\rangle(:)}\textgreater radiusAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the radius property of the disk component.

tfamily \textcolor{red}{\textless double\textgreater} radiusIsGettable() Get the radius property of the disk component.

tfamily \logicalzero radiusIsSettable() Specify whether the radius property of the disk component is settable.

tfamily \void radiusSet(<double> value) Set the radius property of the disk component.

tfamily \void readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezerotationCurve(<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the rotation curve.

tfamily \doublezerotationCurveGradient(<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.
16. Coding GALACTICUS

tfamily \ void serializeValues(<double(>:: array→) Serialize the evolvable quantities to an array.

tfamily \textcolor{red}{\textless textless type(history):\textgreater starFormationHistory()} Get the \textit{starFormationHistory} property of the \textit{disk} component.

tfamily \textcolor{red}{\textless textless type(varying::_string):\textgreater starFormationHistoryAttributeMatch(<logical>[requireGettable→, <logical>[requireSettable→, <logical>[requireEvolvable→) Return a list of implementations that provide the given list of attributes for the \textit{starFormationHistory} property of the \textit{disk} component.

tfamily \intzero starFormationHistoryCount() Compute the count of evolvable quantities in the \textit{starFormationHistory} property of the \textit{DiskExponential} component.

tfamily \textcolor{red}{\textless type(history):\textgreater starFormationHistoryIsGettable()} Get the \textit{starFormationHistory} property of the \textit{disk} component.

tfamily \logicalzero starFormationHistoryIsSettable() Specify whether the \textit{starFormationHistory} property of the \textit{disk} component is settable.

tfamily \ void starFormationHistoryRate(<type(history)> value) Cumulate to the rate of the \textit{starFormationHistory} property of the \textit{DiskExponential} component.

tfamily \ void starFormationHistoryScale(<type(history)> value) Set the scale of the \textit{starFormationHistory} property of the \textit{DiskExponential} component.

tfamily \ void starFormationHistorySet(<type(history)> value) Set the \textit{starFormationHistory} property of the \textit{disk} component.

tfamily \textcolor{red}{\textless double\textgreater starFormationRate()} Get the \textit{starFormationRate} property of the \textit{disk} component.

tfamily \textcolor{red}{\textless textless type(varying::_string):\textgreater starFormationRateAttributeMatch(<logical>[requireGettable→, <logical>[requireSettable→, <logical>[requireEvolvable→) Return a list of implementations that provide the given list of attributes for the \textit{starFormationRate} property of the \textit{disk} component.

tfamily \textcolor{red}{\textless double\textgreater starFormationRateIsGettable()} Get the \textit{starFormationRate} property of the \textit{disk} component.

tfamily \logicalzero starFormationRateIsSettable() Specify whether the \textit{starFormationRate} property of the \textit{disk} component is settable.

tfamily \ textcolor{red}{\textless textless type(history):\textgreater stellarPropertiesHistory()} Get the \textit{stellarPropertiesHistory} property of the \textit{disk} component.

tfamily \textcolor{red}{\textless textless type(varying::_string):\textgreater stellarPropertiesHistoryAttributeMatch(<logical>[requireGettable→, <logical>[requireSettable→, <logical>[requireEvolvable→) Return a list of implementations that provide the given list of attributes for the \textit{stellarPropertiesHistory} property of the \textit{disk} component.

tfamily \intzero stellarPropertiesHistoryCount() Compute the count of evolvable quantities in the \textit{stellarPropertiesHistory} property of the \textit{DiskExponential} component.

tfamily \textcolor{red}{\textless textless type(history):\textgreater stellarPropertiesHistoryIsGettable()} Get the \textit{stellarPropertiesHistory} property of the \textit{disk} component.
16.5. Objects

tfamily \logicalzero stellarPropertiesHistoryIsSettable() Specify whether the stellarPropertiesHistory property of the disk component is settable.

tfamily \void stellarPropertiesHistoryRate(<type(history)> value) Cumulate to the rate of the stellarPropertiesHistory property of the DiskExponential component.

tfamily \void stellarPropertiesHistoryScale(<type(history)> value) Set the scale of the stellarPropertiesHistory property of the DiskExponential component.

tfamily \void stellarPropertiesHistorySet(<type(history)> value) Set the stellarPropertiesHistory property of the disk component.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical→, \langle componentType \rangle \to, \langle massType \rangle \to, \langle weightBy \rangle \to, \langle integer \rangle \to, \langle logical \rangle \to) Compute the surface density.

tfamily \textcolor{red}{\langle type(varying\_string)\rangle} type() Return the type of this object.

tfamily \textcolor{red}{\langle double\rangle} velocity() Get the velocity property of the disk component.

tfamily \textcolor{red}{\langle type(varying\_string)(::)\rangle} velocityAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the velocity property of the disk component.

tfamily \textcolor{red}{\langle double::\rangle} velocityIsGettable() Get the velocity property of the disk component.

tfamily \logicalzero velocityIsSettable() Specify whether the velocity property of the disk component is settable.

tfamily \void velocitySet(<double> value) Set the velocity property of the disk component.

tfamily \logicalzero verySimpleIsActive() Return whether the verySimple implementation of the disk component class is active.

tfamily nodeComponentDynamicsStatistics

tfamily \textcolor{red}{\langle type\rangle} adiabaticRatio() Get the adiabaticRatio property of the dynamicsStatistics component.

tfamily \textcolor{red}{\langle type\rangle} adiabaticRatioAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the adiabaticRatio property of the dynamicsStatistics component.

tfamily \textcolor{red}{\langle type\rangle} adiabaticRatioIsGettable() Get the adiabaticRatio property of the dynamicsStatistics component.

tfamily \logicalzero adiabaticRatioIsSettable() Specify whether the adiabaticRatio property of the dynamicsStatistics component is settable.

tfamily \void adiabaticRatioSet(<double::>) value) Set the adiabaticRatio property of the dynamicsStatistics component.
16. Coding Galacticus

tfamily \textcolor{red}{\textless textless double(:)\textgreater} barInstabilityTimescale() Get the barInstabilityTimescale property of the dynamicsStatistics component.

tfamily \textcolor{red}{\textless textless type(varying\_string)(:)\textgreater} barInstabilityTimescaleAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the barInstabilityTimescale property of the dynamicsStatistics component.

tfamily \textcolor{red}{\textless double(:)\textgreater} barInstabilityTimescaleIsGettable() Get the barInstabilityTimescale property of the dynamicsStatistics component.

tfamily \textcolor{red}{\textless logical\textgreater} barInstabilityTimescaleIsSettable() Specify whether the barInstabilityTimescale property of the dynamicsStatistics component is settable.

tfamily \void barInstabilityTimescaleSet(<double(:)> value) Set the barInstabilityTimescale property of the dynamicsStatistics component.

tfamily \logical zero barsIsActive() Return whether the bars implementation of the dynamicsStatistics component class is active.

tfamily \void builder(<\*type(node)>componentDefinition \rightarrow) Build a nodeComponent from a supplied XML definition.

tfamily \void deserializeRates(<double(:)> array \leftarrow) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array \leftarrow) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array \leftarrow) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle \rightarrow) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \void enclosedMass(<double>radius \rightarrow, <componentType>[componentType] \rightarrow, <massType>[massType] \rightarrow, <weightBy>[weightBy] \rightarrow, <integer>[weightIndex] \rightarrow, <logical>[haloLoaded] \rightarrow) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.
16.5. Objects

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count →, <varying_string> name ←) Return the name of a property given is index.

ntfamily \logicalzero nullIsActive() Return whether the null implementation of the dynamicsStatistics component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer> doubleProperty ↔, <integer doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

ntfamily \void outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

ntfamily \void outputNames(<integer> integerProperty ↔, <char[*](:)> integerPropertyNames ↔, <char[*](:)> integerPropertyComments ↔, <double(:)> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char[*](:)> doublePropertyNames ↔, <char[*](:)> doublePropertyComments ↔, <double(:)> doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

ntfamily \doublezero potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

ntfamily \doublezero rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

ntfamily \doublezero rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

ntfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

ntfamily \void serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

ntfamily \void serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

ntfamily \void serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

ntfamily \doublezero surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the surface density.

ntfamily \textcolor{red}{{\textlessdouble(:)\textgreater}} time() Get the time property of the dynamicsStatistics component.

ntfamily \textcolor{red}{{\textlessvarying\_string(:)\textgreater}} timeAttributeMatch(<logical> [requireGettable] →, <logical> [requireSettable] →, <logical> [requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the time property of the dynamicsStatistics component.

ntfamily \textcolor{red}{{\textlessdouble(:)\textgreater}} timeIsGettable() Get the time property of the dynamicsStatistics component.
tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

tfamily nodeComponentDynamicsStatisticsBars

tfamily \textcolor{red}{\textless double(:)\textgreater} adiabaticRatio() Get the adiabaticRatio property of the dynamicsStatistics component.

tfamily \textcolor{red}{\textless double(:)\textgreater} adiabaticRatioIsGettable() Get the adiabaticRatio property of the dynamicsStatistics component.

tfamily \logicalzero adiabaticRatioIsSettable() Specify whether the adiabaticRatio property of the dynamicsStatistics component is settable.

tfamily \void adiabaticRatioSet(<double(:)> value) Set the adiabaticRatio property of the dynamicsStatistics component.

472

472
tfamily \textbf{\textcolor{red}{\textless *type(treeNode)\textgreater} host()} Return a pointer to the host \textbf{treeNode} object.

tfamily \textbf{\textcolor{red}{\textless *type(treeNode)\textgreater} hotHaloCoolingAbundancesRate(<type(abundances)> value)} Cumulate to the rate of the \textbf{hotHaloCoolingAbundances} property of the \textbf{hotHalo} component.

tfamily \textbf{\textcolor{red}{\textless *type(treeNode)\textgreater} hotHaloCoolingAngularMomentumRate(<double> value)} Cumulate to the rate of the \textbf{hotHaloCoolingAngularMomentum} property of the \textbf{hotHalo} component.

tfamily \textbf{\textcolor{red}{\textless *type(treeNode)\textgreater} hotHaloCoolingMassRate(<double> value)} Cumulate to the rate of the \textbf{hotHaloCoolingMass} property of the \textbf{hotHalo} component.

tfamily \textbf{\textcolor{red}{\textless *type(treeNode)\textgreater} initialize()} Initialize the object.

tfamily \textbf{\textcolor{red}{\textless *type(treeNode)\textgreater} nameFromIndex(<integer> count\rightarrow, \textbf{\textcolor{red}{\textless varying_string\textgreater} name\leftarrow})} Return the name of a property given is index.

tfamily \textbf{\textcolor{red}{\textless *type(treeNode)\textgreater} nullIsActive()} Return whether the null implementation of the \textbf{dynamicsStatistics} component class is active.

tfamily \textbf{\textcolor{red}{\textless *type(treeNode)\textgreater} odeStepRatesInitialize()} Initialize rates for evolvable properties.

tfamily \textbf{\textcolor{red}{\textless *type(treeNode)\textgreater} odeStepScalesInitialize()} Initialize scales for evolvable properties.

tfamily \textbf{\textcolor{red}{\textless *type(treeNode)\textgreater} output(<integer> integerProperty\leftrightarrow, <integer> integerBufferCount\leftrightarrow, <integer(\ldots)> integerBuffer\leftrightarrow, <integer> doubleProperty\leftrightarrow, <integer> doubleBufferCount\leftrightarrow, <double(\ldots)> doubleBuffer\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow)} Generate values of outputtable properties.

tfamily \textbf{\textcolor{red}{\textless *type(treeNode)\textgreater} outputCount(<integer> integerPropertyCount\leftrightarrow, <integer> doublePropertyCount\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow)} Compute a count of outputtable properties.

tfamily \textbf{\textcolor{red}{\textless *type(treeNode)\textgreater} outputNames(<integer> integerProperty\leftrightarrow, <char\[*]\textgreater integerPropertyNames\leftrightarrow, <char\[*]\textgreater(integerPropertyComments\leftrightarrow, <double(\ldots)> integerPropertyUnitsSI\leftrightarrow, <integer> doubleProperty\leftrightarrow, <char\[*]\textgreater doublePropertyNames\leftrightarrow, <char\[*]\textgreater(doublePropertyComments\leftrightarrow, <double(\ldots)> doublePropertyUnitsSI\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow)} Generate names of outputtable properties.

tfamily \textbf{\textcolor{red}{\textless *type(treeNode)\textgreater} potential(<double> radius\rightarrow, <componentType[\ldots]> componentType\rightarrow, <massType[\ldots]> massType\rightarrow, <logical> haloLoaded\rightarrow)} Compute the gravitational potential.
tfamily \textcolor{red}{\textless double(:)\textgreater} adiabaticRatio() Get the adiabaticRatio property of the dynamicsStatistics component.

tfamily \textcolor{red}{\textless double(:)\textgreater} adiabaticRatioAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the adiabaticRatio property of the dynamicsStatistics component

tfamily \textcolor{red}{\textless double(:)\textgreater} adiabaticRatioIsGettable() Get the adiabaticRatioIsGettable() Get the adiabaticRatio property of the dynamicsStatistics component.
16.5. Objects

tfamily logicalzero adiabaticRatioIsSettable() Specify whether the adiabaticRatio property of the dynamicsStatistics component is settable.

tfamily void adiabaticRatioSet(<double()> value) Set the adiabaticRatio property of the dynamicsStatistics component.

tfamily textcolor{red}{textless double():textgreater} barInstabilityTimescale() Get the barInstabilityTimescale property of the dynamicsStatistics component.

tfamily textcolor{red}{textless type(varying\_string):textgreater} barInstabilityTimescaleAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the barInstabilityTimescale property of the dynamicsStatistics component.

tfamily textcolor{red}{textless double():textgreater} barInstabilityTimescaleIsGettable() Get the barInstabilityTimescale property of the dynamicsStatistics component.

tfamily logicalzero barInstabilityTimescaleIsSettable() Specify whether the barInstabilityTimescale property of the dynamicsStatistics component is settable.

tfamily void barInstabilityTimescaleSet(<double()> value) Set the barInstabilityTimescale property of the dynamicsStatistics component.

tfamily logicalzero barsIsActive() Return whether the bars implementation of the dynamicsStatistics component class is active.

tfamily void builder(<*type(node)>componentDefinition) Build a nodeComponent from a supplied XML definition.

tfamily double zero density(<double(3)> positionSpherical→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the density.

tfamily void deserializeRates(<double()> array←) Deserialize the evolvable rates from an array.

tfamily void deserializeScales(<double()> array←) Deserialize the evolvable scales from an array.

tfamily void deserializeValues(<double()> array←) Deserialize the evolvable quantities from an array.

tfamily void destroy() Destroy the object.

tfamily void dump() Generate an ASCII dump of all properties.

tfamily void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily void dumpXML() Generate an XML dump of all properties.

tfamily double zero enclosedMass(<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the mass enclosed within a radius.

tfamily textcolor{red}{textless *type(treeNode):textgreater} host() Return a pointer to the host treeNode object.

tfamily void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.
16. Coding GALACTICUS

tfamily \ void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \ void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \ void initialize() Initialize the object.

tfamily \ void nameFromIndex(<integer> count →, <varying_string> name ←) Return the name of a property given is index.

tfamily \ logicalzero nullsActive() Return whether the null implementation of the dynamicsStatistics component class is active.

tfamily \ void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \ void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \ void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer> doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

tfamily \ void outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily \ void outputNames(<integer> integerProperty ↔, <char[*](:)> integerPropertyNames ↔, <char[*](:)> integerPropertyComments ↔, <double(:)> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char[*](:)> doublePropertyNames ↔, <char[*](:)> doublePropertyComments ↔, <double(:)> doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

tfamily \ doublezero potential(<double> radius →, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

tfamily \ void readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \ doublezero rotationCurve(<double> radius →, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

tfamily \ doublezero rotationCurveGradient(<double> radius →, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

tfamily \ intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \ void serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

tfamily \ void serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

tfamily \ void serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

tfamily \ doublezero surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.

tfamily \ textcolor{red}{\textless double(:)\textgreater} time() Get the time property of the dynamicsStatistics component.
16.5. Objects

tfamily \textcolor{red}{\textless\text{type(varying\_string):}\textgreater} \text{timeAttributeMatch}(<\text{logical}>[\text{requireGettable}]->, <\text{logical}>[\text{requireSettable}]->, <\text{logical}>[\text{requireEvolvable}]->) Return a list of implementations that provide the given list of attributes for the time property of the dynamicsStatistics component.

tfamily \textcolor{red}{\textless\text{double}:\textgreater} \text{timeIsGettable()} Get the time property of the dynamicsStatistics component.

tfamily logicalzero \text{timeIsSettable()} Specify whether the time property of the dynamicsStatistics component is settable.

tfamily void \text{timeSet(<\text{double}:> value)} Set the time property of the dynamicsStatistics component.

tfamily \textcolor{red}{\textless\text{type(varying\_string):}\textgreater} \text{type()} Return the type of this object.

tfamily nodeComponentFormationTime

tfamily void \text{builder(<\text{type(node)}>componentDefinition->)} Build a nodeComponent from a supplied XML definition.

tfamily logicalzero \text{cole2000IsActive()} Return whether the Cole2000 implementation of the formationTime component class is active.

tfamily doublezero \text{density(<\text{double(3)}> positionSpherical->, <\text{componentType}>[\text{componentType}]->, <\text{massType}>[\text{massType}]->, <\text{weightBy}>[\text{weightBy}]->, <\text{integer}>[\text{weightIndex}]->, <\text{logical}>[\text{haloLoaded}]->) Compute the density.

tfamily void \text{deserializeRates(<\text{double}:> array<-)} Deserialize the evolvable rates from an array.

tfamily void \text{deserializeScales(<\text{double}:> array<-)} Deserialize the evolvable scales from an array.

tfamily void \text{deserializeValues(<\text{double}:> array<-)} Deserialize the evolvable quantities from an array.

tfamily void \text{destroy()} Destroy the object.

tfamily void \text{dump()} Generate an ASCII dump of all properties.

tfamily void \text{dumpRaw(<\text{integer}> fileHandle->)} Generate a binary dump of all properties.

tfamily void \text{dumpXML()} Generate an XML dump of all properties.

tfamily doublezero \text{enclosedMass(<\text{double}> radius->, <\text{componentType}>[\text{componentType}]->, <\text{massType}>[\text{massType}]->, <\text{weightBy}>[\text{weightBy}]->, <\text{integer}>[\text{weightIndex}]->, <\text{logical}>[\text{haloLoaded}]->) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless\text{double:}\textgreater} \text{formationTime()} Get the formationTime property of the formationTime component.

tfamily \textcolor{red}{\textless\text{type(varying\_string):}\textgreater} \text{formationTimeAttributeMatch(<\text{logical}>[\text{requireGettable}]->, <\text{logical}>[\text{requireSettable}]->, <\text{logical}>[\text{requireEvolvable}]->) Return a list of implementations that provide the given list of attributes for the formationTime property of the formationTime component.

tfamily \textcolor{red}{\textless\text{double:}\textgreater} \text{formationTimeIsGettable()} Get the formationTime property of the formationTime component.
tfamily \logicalzero formationTimeIsSettable() Specify whether the formationTime property of the formationTime component is settable.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count →, <varying_string>name ←) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the formationTime component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:, :)> integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:, :)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty ↔, <char[ ]*: ]> integerPropertyNames ↔, <char[*]: ]> integerPropertyComments ↔, <double(:, :)> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char[*]: ]> doublePropertyNames ↔, <char[*]: ]> doublePropertyComments ↔, <double(:, :)> doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

tfamily \doublezero potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

tfamily \doublezero rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:, :)> array →) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:, :)> array →) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:, :)> array →) Serialize the evolvable quantities to an array.
16.5. Objects

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical\rightarrow, <componentType> \rightarrow, <massType> \rightarrow, <weightBy> \rightarrow, <integer> \rightarrow, <logical> \rightarrow) Compute the surface density.

tfamily \textcolor{red}\textless type(varying\_string)\textgreater type() Return the type of this object.

tfamily nodeComponentFormationTimeCole2000

tfamily \void builder(<type(node)>componentDefinition\rightarrow) Build a nodeComponent from a supplied XML definition.

tfamily \logicalzero cole2000IsActive() Return whether the Cole2000 implementation of the formation-Time component class is active.

tfamily \doublezero density(<double(3)> positionSpherical\rightarrow, <componentType> \rightarrow, <massType> \rightarrow, <weightBy> \rightarrow, <integer> \rightarrow, <logical> \rightarrow) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle\rightarrow) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<double>radius→, <componentType> \rightarrow, <massType> \rightarrow, <weightBy> \rightarrow, <integer> \rightarrow, <logical> \rightarrow) Compute the mass enclosed within a radius.

tfamily \textcolor{red}\textless double\textgreater formationTime() Get the formationTime property of the formationTime component.

tfamily \textcolor{red}\textless type(treeNode)\textgreater host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.
16. Coding GALACTICUS

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count →, <varying_string> name←) Return the name of a property given is index.

tfamily \logical zero nullIsActive() Return whether the null implementation of the formationTime component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer> doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty↔, <char[*](:)> integerPropertyNames↔, <char[*]( :)> integerPropertyComments↔, <double(:)> integerPropertyUnitsSI↔, <integer> doubleProperty↔, <char[*]( :)> doublePropertyNames↔, <char[*]( :)> doublePropertyComments↔, <double(:)> doublePropertyUnitsSI↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \double zero potential(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

tfamily \void readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \double zero rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

tfamily \double zero rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

tfamily \int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

tfamily \double zero surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.

tfamily \textcolor{red}\{\textless type(varying\_string)\textgreater\} type() Return the type of this object.
16.5. Objects

tfamily nodeComponentFormationTimeNull

\texttt{tfamily void builder(\texttt{<type(node)>componentDefinition}→) Build a nodeComponent from a supplied XML definition.}

\texttt{tfamily logicalzero cole2000IsActive()} Return whether the Cole2000 implementation of the formation-Time component class is active.

\texttt{tfamily doublezero density(\texttt{<double(3)> positionSpherical}→, \texttt{<componentType> [componentType]}→, \texttt{<massType> [massType]}→, \texttt{<weightBy> [weightBy]}→, \texttt{<integer> [weightIndex]}→, \texttt{<logical> [haloLoaded]}→)} Compute the density.

\texttt{tfamily void deserializeRates(\texttt{<double(\_)} array←)} Deserialize the evolvable rates from an array.

\texttt{tfamily void deserializeScales(\texttt{<double(\_)} array←)} Deserialize the evolvable scales from an array.

\texttt{tfamily void deserializeValues(\texttt{<double(\_)} array←)} Deserialize the evolvable quantities from an array.

\texttt{tfamily void destroy()} Destroy the object.

\texttt{tfamily void dump()} Generate an ASCII dump of all properties.

\texttt{tfamily void dumpRaw(\texttt{<integer> fileHandle}→)} Generate a binary dump of all properties.

\texttt{tfamily void dumpXML()} Generate an XML dump of all properties.

\texttt{tfamily doublezero enclosedMass(\texttt{<double> radius}→, \texttt{<componentType> [componentType]}→, \texttt{<massType> [massType]}→, \texttt{<weightBy> [weightBy]}→, \texttt{<integer> [weightIndex]}→, \texttt{<logical> [haloLoaded]}→)} Compute the mass enclosed within a radius.

\texttt{tfamily \textcolor{red}{\textless double\textgreater} formationTime()} Get the formationTime property of the formationTime component.

\texttt{tfamily \textcolor{red}{\textless \textlesstype(varying\_string)(\_)} formationTimeAttributeMatch(\texttt{<logical> [requireGettable]}→, \texttt{<logical> [requireSettable]}→, \texttt{<logical> [requireEvolvable]}→)} Return a list of implementations that provide the given list off attributes for the formationTime property of the formationTime component.

\texttt{tfamily \textcolor{red}{\textless double\textgreater} formationTimeIsGettable()} Get the formationTime property of the formationTime component.

\texttt{tfamily logicalzero formationTimeIsSettable()} Specify whether the formationTime property of the formationTime component is settable.

\texttt{tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host()} Return a pointer to the host treeNode object.

\texttt{tfamily void hotHaloCoolingAbundancesRate(\texttt{<type(abundances)> value}) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.}

\texttt{tfamily void hotHaloCoolingAngularMomentumRate(\texttt{<double> value}) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.}

\texttt{tfamily void hotHaloCoolingMassRate(\texttt{<double> value}) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.}

\texttt{tfamily void initialize()} Initialize the object.
tfamily \void nameFromIndex(<integer> count\rightarrow, <varying_string>name\leftarrow) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the formationTime component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty\leftrightarrow, <integer> integerBufferCount\leftrightarrow, <integer(\ldots)> integerBuffer\leftrightarrow, <integer>doubleProperty\leftrightarrow, <integer>doubleBufferCount\leftrightarrow, <double(\ldots)> doubleBuffer\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow) Generate values of outputtable properties.

tfamily \void outputCount(<integer>integerPropertyCount\leftrightarrow, <integer>doublePropertyCount\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow) Compute a count of outputtable properties.

tfamily \void outputNames(<integer>integerProperty\leftrightarrow, <char[*](\ldots)>integerPropertyNames\leftrightarrow, <char[*](\ldots)>integerPropertyComments\leftrightarrow, <double(\ldots)>integerPropertyUnitsSI\leftrightarrow, <integer>doubleProperty\leftrightarrow, <char[*](\ldots)>doublePropertyNames\leftrightarrow, <char[*](\ldots)>doublePropertyComments\leftrightarrow, <double(\ldots)>doublePropertyUnitsSI\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow) Generate names of outputtable properties.

tfamily \doublezero potential(<double> radius\rightarrow, <componentType>[componentType]\rightarrow, <massType>[massType]\rightarrow, <logical>[haloLoaded]\rightarrow) Compute the gravitational potential.

tfamily \void readRaw(<integer> fileHandle\rightarrow) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezero rotationCurve(<double> radius\rightarrow, <componentType>[componentType]\rightarrow, <massType>[massType]\rightarrow, <logical>[haloLoaded]\rightarrow) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double> radius\rightarrow, <componentType>[componentType]\rightarrow, <massType>[massType]\rightarrow, <logical>[haloLoaded]\rightarrow) Compute the rotation curve gradient.

tfamily \into serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(\ldots)> array\rightarrow) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(\ldots)> array\rightarrow) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(\ldots)> array\rightarrow) Serialize the evolvable quantities to an array.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical\rightarrow, <componentType>[componentType]\rightarrow, <massType>[massType]\rightarrow, <weightBy>[weightBy]\rightarrow, <integer>[weightIndex]\rightarrow, <logical>[haloLoaded]\rightarrow) Compute the surface density.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.
16.5. Objects

tfamily nodeComponentHostHistory

tfamily \ void builder(<%type(node)>componentDefinition\to) Build a nodeComponent from a supplied XML definition.

tfamily \ double zero density(<%double(3)>positionSpherical\to, <%componentType>[componentType]\to, <%massType>[massType]\to, <%weightBy>[weightBy]\to, <%integer>[weightIndex]\to, <%logical>[haloLoaded]\to) Compute the density.

tfamily \ void deserializeRates(<%double(\to)) Deserialize the evolvable rates from an array.

tfamily \ void deserializeScales(<%double(\to)) Deserialize the evolvable scales from an array.

tfamily \ void deserializeValues(<%double(\to)) Deserialize the evolvable quantities from an array.

tfamily \ void destroy() Destroy the object.

tfamily \ void dump() Generate an ASCII dump of all properties.

tfamily \ void dumpRaw(<%integer>fileHandle\to) Generate a binary dump of all properties.

tfamily \ void dumpXML() Generate an XML dump of all properties.

tfamily \ double zero enclosedMass(<%double>radius\to, <%componentType>[componentType]\to, <%massType>[massType]\to, <%weightBy>[weightBy]\to, <%integer>[weightIndex]\to, <%logical>[haloLoaded]\to) Compute the mass enclosed within a radius.

tfamily \ textcolor{red}{\textless type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \ textcolor{red}{\textless double\textgreater} hostMassMaximum() Get the hostMassMaximum property of the hostHistory component.

tfamily \ textcolor{red}{\textless type(varying\_string)(\to)} hostMassMaximumAttributeMatch(<%logical>[requireGettable]\to, <%logical>[requireSettable]\to, <%logical>[requireEvolvable]\to) Return a list of implementations that provide the given list off attributes for the hostMassMaximum property of the hostHistory component.

tfamily \ textcolor{red}{\textless double\textgreater} hostMassMaximumIsGettable() Get the hostMassMaximum property of the hostHistory component.

tfamily \ logical zero hostMassMaximumIsSettable() Specify whether the hostMassMaximum property of the hostHistory component is settable.

tfamily \ void hostMassMaximumSet(<%double>value) Set the hostMassMaximum property of the hostHistory component.

tfamily \ void hotHaloCoolingAbundancesRate(<%type(abundances)>value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \ void hotHaloCoolingAngularMomentumRate(<%double>value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \ void hotHaloCoolingMassRate(<%double>value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \ void initialize() Initialize the object.
tfamily \void nameFromIndex (<integer> count \rightarrow, <varying_string> name \leftarrow) Return the name of a property given is index.

tfamily \logical zero nullIsActive() Return whether the null implementation of the hostHistory component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output (<integer> integerProperty \leftrightarrow, <integer> integerBufferCount \leftrightarrow, <integer(::)> integerBuffer \leftrightarrow, <integer> doubleProperty \leftrightarrow, <integer> doubleBufferCount \leftrightarrow, <double(::)> doubleBuffer \leftrightarrow, <double> time \rightarrow, <integer> instance \rightarrow) Generate values of outputtable properties.

tfamily \void outputCount (<integer> integerPropertyCount \leftrightarrow, <integer> doublePropertyCount \leftrightarrow, <double> time \rightarrow, <integer> instance \rightarrow) Compute a count of outputtable properties.

tfamily \void outputNames (<integer> integerProperty \leftrightarrow, <char[*]> integerPropertyNames \leftrightarrow, <char[*]> integerPropertyComments \leftrightarrow, <double(::)> integerPropertyUnitsSI \leftrightarrow, <integer> doubleProperty \leftrightarrow, <char[*]> doublePropertyNames \leftrightarrow, <char[*]> doublePropertyComments \leftrightarrow, <double(::)> doublePropertyUnitsSI \leftrightarrow, <double> time \rightarrow, <integer> instance \rightarrow) Generate names of outputtable properties.

tfamily \double zero potential (<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the gravitational potential.

tfamily \double zero rotationCurve (<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the rotation curve.

tfamily \double zero rotationCurveGradient (<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the rotation curve gradient.

tfamily \int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates (<double(::)> array \rightarrow) Serialize the evolvable rates to an array.

tfamily \void serializeScales (<double(::)> array \rightarrow) Serialize the evolvable scales to an array.

tfamily \void serializeValues (<double(::)> array \rightarrow) Serialize the evolvable quantities to an array.

tfamily \logical zero standardIsActive() Return whether the standard implementation of the hostHistory component class is active.

tfamily \double zero surfaceDensity (<double(3)> positionCylindrical \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <weightBy> [weightBy] \rightarrow, <integer> [weightIndex] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the surface density.

tfamily \textcolor{red}\textless \texttt{type(varying_string)}\textgreater\ type() Return the type of this object.
16.5. Objects

tfamily nodeComponentHostHistoryNull

tfamily \ void builder(\type(node)\ componentDefinition\ →) Build a \nodeComponent\ from a supplied \XML\ definition.

tfamily \ double zero \ density(\double(3)\ position\ spherical\ →, \componentType\ \componentType\ →, \massType\ \massType\ →, \weightBy\ \weightBy\ →, \integer\ \integer\ →, \logical\ \haloLoaded\ →) Compute the density.

tfamily \ void \ deserializeRates(\double(:)\ array←) Deserialize the evolvable rates from an array.

tfamily \ void \ deserializeScales(\double(:)\ array←) Deserialize the evolvable scales from an array.

tfamily \ void \ deserializeValues(\double(:)\ array←) Deserialize the evolvable quantities from an array.

tfamily \ void \ destroy() Destroy the object.

tfamily \ void \ dump() Generate an ASCII dump of all properties.

tfamily \ void \ dumpRaw(\integer\ fileHandle→) Generate a binary dump of all properties.

tfamily \ void \ dumpXML() Generate an XML dump of all properties.

tfamily \ double zero \ enclosedMass(\double\ radius→, \componentType\ \componentType\ →, \massType\ \massType\ →, \weightBy\ \weightBy\ →, \integer\ \integer\ →, \logical\ \haloLoaded\ →) Compute the mass enclosed within a radius.

tfamily \ textual {red} \ type(treeNode)\ host() Return a pointer to the host \treeNode\ object.

tfamily \ textual {red} \ hostMassMaximum() Get the hostMassMaximum property of the hostHistory component.

tfamily \ textual {red} \ hostMassMaximum\AttributeMatch(\logical\ \requireGettable→, \logical\ \requireSettable→, \logical\ \requireEvolvable→) Return a list of implementations that provide the given list of attributes for the hostMassMaximum property of the hostHistory component.

tfamily \ textual {red} \ hostMassMaximum\IsGettable() Get the hostMassMaximum property of the hostHistory component.

tfamily \ logical zero \ hostMassMaximum\IsSettable() Specify whether the hostMassMaximum property of the hostHistory component is settable.

tfamily \ void \ hostMassMaximum\Set(\double\ value) Set the hostMassMaximum property of the hostHistory component.

tfamily \ void \ hotHaloCooling\AbundancesRate(\type(abundances)\ value) Cumulate to the rate of the hotHaloCooling\Abundances property of the hotHalo component.

tfamily \ void \ hotHaloCooling\AngularMomentumRate(\double\ value) Cumulate to the rate of the hotHaloCooling\AngularMomentum property of the hotHalo component.

tfamily \ void \ hotHaloCooling\MassRate(\double\ value) Cumulate to the rate of the hotHaloCooling\Mass property of the hotHalo component.

tfamily \ void \ initialize() Initialize the object.
tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.
16.5. Objects

tfamily nodeComponentHostHistoryStandard

tfamily \void builder(<\texttt{type(node)}\texttt{componentDefinition}\rightarrow) Build a nodeComponent from a supplied XML definition.

tfamily \doublezero density(<\texttt{double(3)}\texttt{positionSpherical}\rightarrow, <\texttt{componentType}\texttt{componentType}\rightarrow, <\texttt{massType}\texttt{massType}\rightarrow, <\texttt{weightBy}\texttt{weightBy}\rightarrow, <\texttt{integer}\texttt{weightIndex}\rightarrow, <\texttt{logical}\texttt{haloLoaded}\rightarrow) Compute the density.

tfamily \void deserializeRates(<\texttt{double(:)}\texttt{array}\leftarrow) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<\texttt{double(:)}\texttt{array}\leftarrow) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<\texttt{double(:)}\texttt{array}\leftarrow) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<\texttt{integer}\texttt{fileHandle}\rightarrow) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<\texttt{double}\texttt{radius}\rightarrow, <\texttt{componentType}\texttt{componentType}\rightarrow, <\texttt{massType}\texttt{massType}\rightarrow, <\texttt{weightBy}\texttt{weightBy}\rightarrow, <\texttt{integer}\texttt{weightIndex}\rightarrow, <\texttt{logical}\texttt{haloLoaded}\rightarrow) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless\texttt{type(treeNode)}\textgreater} host() Return a pointer to the host treeNode object.

tfamily \textcolor{red}{\textless\texttt{double}\textgreater} hostMassMaximum() Get the hostMassMaximum property of the hostHistory component.

tfamily \textcolor{red}{\textless\texttt{type(varying\_string)}\textgreater} hostMassMaximumAttributeMatch(<\texttt{logical}\texttt{requireGettable}\rightarrow, <\texttt{logical}\texttt{requireSettable}\rightarrow, <\texttt{logical}\texttt{requireEvolvable}\rightarrow) Return a list of implementations that provide the given list off attributes for the hostMassMaximum property of the hostHistory component.

tfamily \textcolor{red}{\textless\texttt{double}\textgreater} hostMassMaximumIsGettable() Get the hostMassMaximum property of the hostHistory component.

tfamily \logicalzero hostMassMaximumIsSettable() Specify whether the hostMassMaximum property of the hostHistory component is settable.

tfamily \void hostMassMaximumSet(<\texttt{double}\texttt{value}) Set the hostMassMaximum property of the hostHistory component.

tfamily \void hotHaloCoolingAbundancesRate(<\texttt{type(abundances)}\texttt{value}) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<\texttt{double}\texttt{value}) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<\texttt{double}\texttt{value}) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.
tfamily \void nameFromIndex(<integer> count\rightarrow, <varying_string> name\leftarrow) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the hostHistory component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer> doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty↔, <char[*](:)> integerPropertyNames↔, <char[*](:)> integerPropertyComments↔, <double(:)> integerPropertyUnitsSI↔, <integer> doubleProperty↔, <char[*](:)> doublePropertyNames↔, <char[*](:)> doublePropertyComments↔, <double(:)> doublePropertyUnitsSI↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \doublezero potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

tfamily \void readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezero rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the hostHistory component class is active.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.

tfamily \textcolor{red}{\textless type(varying}_{\_}\_string}\textgreater type() Return the type of this object.
16.5. Objects

```
tfamily nodeComponentHotHalo

  tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundances() Get the abundances property of the hotHalo component.

  tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} abundancesAttributeMatch(<logical>[requireGettable] → <logical>[requireSettable] → <logical>[requireEvolvable] → ) Return a list of implementations that provide the given list of attributes for the abundances property of the hotHalo component.

  tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesCold() Get the abundancesCold property of the hotHalo component.

  tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} abundancesColdAttributeMatch(<logical>[requireGettable] → <logical>[requireSettable] → <logical>[requireEvolvable] → ) Return a list of implementations that provide the given list of attributes for the abundancesCold property of the hotHalo component.

  tfamily \intzero abundancesColdCount() Compute the count of evolvable quantities in the abundancesCold property of the HotHaloColdMode component.

  tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesColdIsGettable() Get the abundancesCold property of the hotHalo component.

  tfamily \logical zero abundancesColdIsSettable() Specify whether the abundancesCold property of the hotHalo component is settable.

  tfamily \void abundancesColdRate(<type(abundances)> value) Cumulate to the rate of the abundancesCold property of the HotHaloColdMode component.

  tfamily \void abundancesColdScale(<type(abundances)> value) Set the scale of the abundancesCold property of the HotHaloColdMode component.

  tfamily \void abundancesColdSet(<type(abundances)> value) Set the abundancesCold property of the hotHalo component.

  tfamily \intzero abundancesCount() Compute the count of evolvable quantities in the abundances property of the HotHaloStandard component.

  tfamily \textcolor{red}{\textless double\textgreater} angularMomentum() Get the angularMomentum property of the hotHalo component.

  tfamily \logical zero abundancesIsGettable() Specify whether the abundances property of the hotHalo component is settable.

  tfamily \void abundancesRate(<type(abundances)> value) Cumulate to the rate of the abundances property of the HotHaloStandard component.

  tfamily \void abundancesScale(<type(abundances)> value) Set the scale of the abundances property of the HotHaloStandard component.

  tfamily \void abundancesSet(<type(abundances)> value) Set the abundances property of the hotHalo component.

  tfamily \textcolor{red}{\textless double\textgreater} angularMomentum() Get the angularMomentum property of the hotHalo component.
```
tfamily \textcolor{red}{\textless double\textgreater} angularMomentumAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the angularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumCold() Get the angularMomentumCold property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumColdAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the angularMomentumCold property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumColdCount() Compute the count of evolvable quantities in the angularMomentumCold property of the HotHaloColdMode component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumColdIsGettable() Get the angularMomentumCold property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumColdIsSettable() Specify whether the angularMomentumCold property of the hotHalo component is settable.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumColdRate(<double> value) Cumulate to the rate of the angularMomentumCold property of the HotHaloColdMode component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumColdScale(<double> value) Set the scale of the angularMomentumCold property of the HotHaloColdMode component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumColdSet(<double> value) Set the angularMomentumCold property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumIsGettable() Get the angularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumIsSettable() Specify whether the angularMomentum property of the hotHalo component is settable.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumSet(<double> value) Set the angularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily \textcolor{red}{\textless double\textgreater} chemicals() Get the chemicals property of the hotHalo component.
16.5. Objects

tfamily \textcolor{red}{\textless\textless varying\_string\rangle\textless\textgreater} chemicalsAttributeMatch(\langle\text{logical}\rangle[\text{requireGettable}]
\rightarrow, \langle\text{logical}\rangle[\text{requireSettable}]
\rightarrow, \langle\text{logical}\rangle[\text{requireEvolvable}]
\rightarrow) Return a list of implementations that provide the given list of attributes for the chemicals property of the hotHalo component.

tfamily \intzero chemicalsCount() Compute the count of evolvable quantities in the chemicals property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textless chemicalAbundances\rangle\textless\textgreater} chemicalsIsGettable() Get the chemicals property of the hotHalo component.

tfamily \logicalzero chemicalsIsSettable() Specify whether the chemicals property of the hotHalo component is settable.

tfamily \void chemicalsRate(\langle\text{type(chemicalAbundances)}\rangle\text{value}) Cumulate to the rate of the chemicals property of the HotHaloStandard component.

tfamily \void chemicalsScale(\langle\text{type(chemicalAbundances)}\rangle\text{value}) Set the scale of the chemicals property of the HotHaloStandard component.

tfamily \void chemicalsSet(\langle\text{type(chemicalAbundances)}\rangle\text{value}) Set the chemicals property of the hotHalo component.

tfamily \logicalzero coldModeIsActive() Return whether the coldMode implementation of the hotHalo component class is active.

tfamily \doublezero density(\langle\text{double(3)}\rangle\text{positionSpherical}\rightarrow, \langle\text{componentType}\rangle[\text{componentType}]\rightarrow,
\langle\text{massType}\rangle[\text{massType}]\rightarrow, \langle\text{weightBy}\rangle[\text{weightBy}]\rightarrow, \langle\text{integer}\rangle[\text{weightIndex}]\rightarrow, \langle\text{logical}\rangle[\text{haloLoaded}]\rightarrow) Compute the density.

tfamily \void deserializeRates(\langle\text{double(\_)}\rangle\text{array}←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(\langle\text{double(\_)}\rangle\text{array}←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(\langle\text{double(\_)}\rangle\text{array}←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(\langle\text{integer}\rangle\text{fileHandle}→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(\langle\text{double}\rangle\text{radius}→, \langle\text{componentType}\rangle[\text{componentType}]\rightarrow,
\langle\text{massType}\rangle[\text{massType}]\rightarrow, \langle\text{weightBy}\rangle[\text{weightBy}]→, \langle\text{integer}\rangle[\text{weightIndex}]→, \langle\text{logical}\rangle[\text{haloLoaded}]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless\textless double\textgreater} heatSource() Get the heatSource property of the hotHalo component.

tfamily \textcolor{red}{\textless\textless varying\_string\rangle\textless\textgreater} heatSourceAttributeMatch(\langle\text{logical}\rangle[\text{requireGettable}]
\rightarrow, \langle\text{logical}\rangle[\text{requireSettable}]
\rightarrow, \langle\text{logical}\rangle[\text{requireEvolvable}]
\rightarrow) Return a list of implementations that provide the given list of attributes for the heatSource property of the hotHalo component.

tfamily \intzero heatSourceCount() Compute the count of evolvable quantities in the heatSource property of the HotHaloStandard component.
tfamily \textcolor{red}{\textless double\textgreater} heatSourceIsGettable() Get the \texttt{heatSource} property of the \texttt{hotHalo} component.

tfamily \textcolor{red}{\textlesstype(abundances)\textgreater} hotHaloCoolingAbundancesIsGettable() Get the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component.

tfamily \logicalzero heatSourceIsSettable() Specify whether the \texttt{heatSource} property of the \texttt{hotHalo} component is settable.

tfamily \void heatSourceRate(<double> value) Cumulate to the rate of the \texttt{heatSource} property of the \texttt{HotHaloStandard} component.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host \texttt{treeNode} object.

tfamily \textcolor{red}{\textlesstype(\_string)(:)\textgreater} hotHaloCoolingAbundancesAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component.

tfamily \intzero hotHaloCoolingAbundancesCount() Compute the count of evolvable quantities in the \texttt{hotHaloCoolingAbundances} property of the \texttt{HotHaloStandard} component.

tfamily \textcolor{red}{\textless double\textgreater} hotHaloCoolingAngularMomentum() Get the \texttt{hotHaloCoolingAngularMomentum} property of the \texttt{hotHalo} component.

tfamily \textcolor{red}{\textlesstype(\_string)(:)\textgreater} hotHaloCoolingAngularMomentumAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the \texttt{hotHaloCoolingAngularMomentum} property of the \texttt{hotHalo} component.

tfamily \intzero hotHaloCoolingAngularMomentumCount() Compute the count of evolvable quantities in the \texttt{hotHaloCoolingAngularMomentum} property of the \texttt{HotHaloStandard} component.
tfamily \ void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \ void hotHaloCoolingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \ logical zero hotHaloCoolingAngularMomentumRateIsAttached() Return whether the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component has been attached to a function.

tfamily \ textcolor{red}{\textless double\textgreater} hotHaloCoolingMass() Get the hotHaloCoolingMass property of the hotHalo component.

tfamily \ textcolor{red}{\textless logical\textgreater} hotHaloCoolingMassIsGettable() Get the hotHaloCoolingMass property of the hotHalo component.

tfamily \ void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \ void hotHaloCoolingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingMass property of the hotHalo component.

tfamily \ logical zero hotHaloCoolingMassRateIsAttached() Return whether the rate method of the hotHaloCoolingMass property of the hotHalo component has been attached to a function.

tfamily \ void initialize() Initialize the object.

tfamily \ textcolor{red}{\textless logical\textgreater} isInitialized() Get the isInitialized property of the hotHalo component.

tfamily \ textcolor{red}{\textless logical\textgreater} isInitializedIsGettable() Get the isInitialized property of the hotHalo component.

tfamily \ void isInitializedSet(<logical> value) Set the isInitialized property of the hotHalo component.

493
16. Coding GALACTICUS

tfamily \textcolor{red}\{\textless double\textgreater\} mass() Get the mass property of the hotHalo component.

tfamily \textcolor{red}\{\textless double\textgreater\} massAttributeMatch(<logical>[requireGettable]\rightarrow, <logical>[requireSettable]\rightarrow, <logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list off attributes for the mass property of the hotHalo component.

tfamily \textcolor{red}\{\textless double\textgreater\} massCold() Get the massCold property of the hotHalo component.

tfamily \textcolor{red}\{\textless double\textgreater\} massColdAttributeMatch(<logical>[requireGettable]\rightarrow, <logical>[requireSettable]\rightarrow, <logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list off attributes for the massCold property of the hotHalo component.

tfamily \intzero massColdCount() Compute the count of evolvable quantities in the massCold property of the HotHaloColdMode component.

tfamily \textcolor{red}\{\textless double\textgreater\} massColdIsGettable() Get the massCold property of the hotHalo component.

tfamily \logicalzero massColdIsSettable() Specify whether the massCold property of the hotHalo component is settable.

tfamily \void massColdRate(<double> value) Cumulate to the rate of the massCold property of the HotHaloColdMode component.

tfamily \void massColdScale(<double> value) Set the scale of the massCold property of the HotHaloColdMode component.

tfamily \void massColdSet(<double> value) Set the massCold property of the hotHalo component.

tfamily \intzero massCount() Compute the count of evolvable quantities in the mass property of the HotHaloStandard component.

tfamily \textcolor{red}\{\textless double\textgreater\} massIsGettable() Get the mass property of the hotHalo component.

tfamily \logicalzero massIsSettable() Specify whether the mass property of the hotHalo component is settable.

tfamily \void massRate(<double> value) Cumulate to the rate of the mass property of the HotHaloStandard component.

tfamily \void massScale(<double> value) Set the scale of the mass property of the HotHaloStandard component.

tfamily \void massSet(<double> value) Set the mass property of the hotHalo component.

tfamily \textcolor{red}\{\textless double\textgreater\} massSink() Get the massSink property of the hotHalo component.

tfamily \textcolor{red}\{\textless double\textgreater\} massSinkAttributeMatch(<logical>[requireGettable]\rightarrow, <logical>[requireSettable]\rightarrow, <logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list off attributes for the massSink property of the hotHalo component.
tfamily \textcolor{red}{\textless double\textgreater} massSinkIsGettable() Get the massSink property of the hotHalo component.

tfamily \logicalzero massSinkIsSettable() Specify whether the massSink property of the hotHalo component is settable.

tfamily \void massSinkRate(<double> value) Cumulate to the rate of the massSink property of the HotHaloStandard component.

tfamily \void nameFromIndex(<integer> count →, <varying_string>name←) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the hotHalo component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \textcolor{red}{\textless double\textgreater} outerRadius() Get the outerRadius property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} outflowedAbundances() Get the outflowedAbundances property of the hotHalo component.

tfamily \void outerRadiusScale(<double> value) Set the scale of the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusSet(<double> value) Set the outerRadius property of the hotHalo component.
16. Coding GALACTICUS

tfamily \intzero outflowedAbundancesCount() Compute the count of evolvable quantities in the outflowedAbundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textless\texttxtless\type(abundances)\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater} outflowedAbundancesIsGettable() Get the outflowedAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype(abundances)\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater} outflowedAbundancesIsSettable() Specify whether the outflowedAbundances property of the hotHalo component is settable.

tfamily \void outflowedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the outflowedAbundances property of the HotHaloStandard component.

tfamily \void outflowedAbundancesScale(<type(abundances)> value) Set the scale of the outflowedAbundances property of the HotHaloStandard component.

tfamily \void outflowedAbundancesSet(<type(abundances)> value) Set the outflowedAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype\textgreater\textgreater} outflowedAngularMomentum() Get the outflowedAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater} outflowedAngularMomentumAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the outflowedAngularMomentum property of the hotHalo component.

tfamily \intzero outflowedAngularMomentumCount() Compute the count of evolvable quantities in the outflowedAngularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textlesstype\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater} outflowedAngularMomentumIsGettable() Get the outflowedAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater} outflowedAngularMomentumIsSettable() Specify whether the outflowedAngularMomentum property of the hotHalo component is settable.

tfamily \void outflowedAngularMomentumRate(<double> value) Cumulate to the rate of the outflowedAngularMomentum property of the HotHaloStandard component.

tfamily \void outflowedAngularMomentumScale(<double> value) Set the scale of the outflowedAngularMomentum property of the HotHaloStandard component.

tfamily \void outflowedAngularMomentumSet(<double> value) Set the outflowedAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater} outflowedMass() Get the outflowedMass property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater} outflowedMassAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the outflowedMass property of the hotHalo component.

tfamily \intzero outflowedMassCount() Compute the count of evolvable quantities in the outflowedMass property of the HotHaloStandard component.

tfamily \textcolor{red}{\textlesstype\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater\textgreater} outflowedMassIsGettable() Get the outflowedMass property of the hotHalo component.
16.5. Objects

tfamily \logicalzero outflowedMassIsSettable() Specify whether the \textcolor{red}{\textlesstype(abundances)\textgreater} outflowedMass property of the \textcolor{red}{\textlesstype(varying\_string)(:)\textgreater} hotHalo component is settable.

tfamily \void outflowedMassRate(<\text{double}> value) Cumulate to the rate of the \textcolor{red}{\textlesstype(abundances)\textgreater} outflowedMass property of the HotHaloStandard component.

tfamily \void outflowedMassScale(<\text{double}> value) Set the scale of the \textcolor{red}{\textlesstype(abundances)\textgreater} outflowedMass property of the HotHaloStandard component.

tfamily \void outflowedMassSet(<\text{double}> value) Set the \textcolor{red}{\textlesstype(abundances)\textgreater} outflowedMass property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype(abundances)\textgreater} outflowingAbundances() Get the \textcolor{red}{\textlesstype(varying\_string)(:)\textgreater} outflowingAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)\textgreater} outflowingAbundancesAttributeMatch(<\logical>[requireGettable]→, <\logical>[requireSettable]→, <\logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the \textcolor{red}{\textlesstype(abundances)\textgreater} outflowingAbundances property of the hotHalo component.

tfamily \intzero outflowingAbundancesCount() Compute the count of evolvable quantities in the \textcolor{red}{\textlesstype(varying\_string)(:)\textgreater} outflowingAbundances property of the HotHaloStandard component.

tfamily \logicalzero outflowingAbundancesIsSettable() Specify whether the \textcolor{red}{\textlesstype(abundances)\textgreater} outflowingAbundances property of the hotHalo component is settable.

tfamily \void outflowingAbundancesRate(<\text{type(abundances)}> value) Cumulate to the rate of the \textcolor{red}{\textlesstype(abundances)\textgreater} outflowingAbundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textlesstype(abundances)\textgreater} outflowingAngularMomentum() Get the \textcolor{red}{\textlesstype(varying\_string)(:)\textgreater} outflowingAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)\textgreater} outflowingAngularMomentumAttributeMatch(<\logical>[requireGettable]→, <\logical>[requireSettable]→, <\logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the \textcolor{red}{\textlesstype(abundances)\textgreater} outflowingAngularMomentum property of the hotHalo component.

tfamily \intzero outflowingAngularMomentumCount() Compute the count of evolvable quantities in the \textcolor{red}{\textlesstype(abundances)\textgreater} outflowingAngularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textlesstype(abundances)\textgreater} outflowingAngularMomentumIsGettable() Get the \textcolor{red}{\textlesstype(abundances)\textgreater} outflowingAngularMomentum property of the hotHalo component.

tfamily \logicalzero outflowingAngularMomentumIsSettable() Specify whether the \textcolor{red}{\textlesstype(abundances)\textgreater} outflowingAngularMomentum property of the hotHalo component is settable.

tfamily \void outflowingAngularMomentumRate(<\text{double}> value) Cumulate to the rate of the \textcolor{red}{\textlesstype(abundances)\textgreater} outflowingAngularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textlesstype(abundances)\textgreater} outflowingMass() Get the \textcolor{red}{\textlesstype(abundances)\textgreater} outflowingMass property of the hotHalo component.
tfamily \textcolor{red}{\textless\textgreater}

\textlesstype(outflowingMassAttributeMatch(<\textlesstype[requireGettable]→,<\textlesstype[requireSettable]→,<\textlesstype[requireEvolvable]→))

Return a list of implementations that provide the given list of attributes for the \textlesstype(outflowingMass) property of the hotHalo component.

tfamily \intzero

\textlesstype(outflowingMassCount())

Compute the count of evolvable quantities in the \textlesstype(outflowingMass) property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textgreater}

\textlesstype(outflowingMassIsGettable())

Get the \textlesstype(outflowingMass) property of the hotHalo component.

tfamily \logicalzero

\textlesstype(outflowingMassIsSettable())

Specify whether the \textlesstype(outflowingMass) property of the hotHalo component is settable.

tfamily \void

\textlesstype(outflowingMassRate(<\textlesstype[double] value))

Cumulate to the rate of the \textlesstype(outflowingMass) property of the HotHaloStandard component.

tfamily \logicalzero

\textlesstype(outflowTrackingIsActive())

Return whether the outflowTracking implementation of the hotHalo component class is active.

tfamily \void

\textlesstype(output(<\textlesstype[integer] integerProperty↔,<\textlesstype[integer] integerBufferCount↔,<\textlesstype[integer] integerBuffer↔,<\textlesstype[integer] doubleProperty↔,<\textlesstype[integer] doubleBufferCount↔,<\textlesstype[double] time→,<\textlesstype[integer] instance→)))

Generate values of outputtable properties.

tfamily \void

\textlesstype(outputCount(<\textlesstype[integer] integerPropertyCount↔,<\textlesstype[integer] doublePropertyCount↔,<\textlesstype[double] time→,<\textlesstype[integer] instance→))

Compute a count of outputtable properties.

tfamily \void

\textlesstype(outputNames(<\textlesstype[integer] integerProperty↔,<\textlesstype[char\[*\]() integerPropertyNames↔,<\textlesstype[char\[*\]() integerPropertyComments↔,<\textlesstype[double] integerPropertyUnitsSI↔,<\textlesstype[integer] doubleProperty↔,<\textlesstype[char\[*\]() doublePropertyNames↔,<\textlesstype[char\[*\]() doublePropertyComments↔,<\textlesstype[double] doublePropertyUnitsSI↔,<\textlesstype[double] time→,<\textlesstype[integer] instance→))

Generate names of outputtable properties.

tfamily \doublezero

\textlesstype(potential(<\textlesstype[double] radius→,<\textlesstype[componentType] componentType→,<\textlesstype[massType] massType→,<\textlesstype[logical] haloLoaded→))

Compute the gravitational potential.

tfamily \doublezero

\textlesstype(rotationCurve(<\textlesstype[double] radius→,<\textlesstype[componentType] componentType→,<\textlesstype[massType] massType→,<\textlesstype[logical] haloLoaded→))

Compute the rotation curve.

tfamily \doublezero

\textlesstype(rotationCurveGradient(<\textlesstype[double] radius→,<\textlesstype[componentType] componentType→,<\textlesstype[massType] massType→,<\textlesstype[logical] haloLoaded→))

Compute the rotation curve gradient.

tfamily \intzero

\textlesstype(serializeCount())

Return a count of the number of evolvable quantities to be evolved.

tfamily \void

\textlesstype(serializeRates(<\textlesstype[double] array→))

Serialize the evolvable rates to an array.

tfamily \void

\textlesstype(serializeScales(<\textlesstype[double] array→))

Serialize the evolvable scales to an array.

tfamily \logicalzero

\textlesstype(serializeValues(<\textlesstype[double] array→))

Serialize the evolvable quantities to an array.

tfamily \logicalzero

\textlesstype(standardIsActive())

Return whether the standard implementation of the hotHalo component class is active.

tfamily \textcolor{red}{\textlesstype(abundances)\textless\textgreater}

\textlesstype(strippedAbundances())

Get the \textlesstype(strippedAbundances) property of the hotHalo component.
16.5. Objects

tfamily \textcolor{red}{\textless type(varying\_string):\textgreater} strippedAbundancesAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the strippedAbundances property of the hotHalo component.

tfamily \intzero strippedAbundancesCount() Compute the count of evolvable quantities in the strippedAbundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless type(abundances):\textgreater} strippedAbundancesIsGettable() Get the strippedAbundances property of the hotHalo component.

tfamily \logicalzero strippedAbundancesIsSettable() Specify whether the strippedAbundances property of the hotHalo component is settable.

tfamily \void strippedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the strippedAbundances property of the HotHaloStandard component.

tfamily \void strippedAbundancesScale(<type(abundances)> value) Set the scale of the strippedAbundances property of the HotHaloStandard component.

tfamily \void strippedAbundancesSet(<type(abundances)> value) Set the strippedAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless double:\textgreater} strippedMass() Get the strippedMass property of the hotHalo component.

tfamily \textcolor{red}{\textless type(abundances):\textgreater} strippedMassAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the strippedMass property of the hotHalo component.

tfamily \intzero strippedMassCount() Compute the count of evolvable quantities in the strippedMass property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double:\textgreater} strippedMassIsGettable() Get the strippedMass property of the hotHalo component.

tfamily \logicalzero strippedMassIsSettable() Specify whether the strippedMass property of the hotHalo component is settable.

tfamily \void strippedMassRate(<double> value) Cumulate to the rate of the strippedMass property of the HotHaloStandard component.

tfamily \void strippedMassScale(<double> value) Set the scale of the strippedMass property of the HotHaloStandard component.

tfamily \void strippedMassSet(<double> value) Set the strippedMass property of the hotHalo component.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the surface density.

tfamily \textcolor{red}{\textless type(abundances):\textgreater} trackedOutflowAbundances() Get the trackedOutflowAbundances property of the hotHalo component.
16. Coding Galacticus

tfamily \textcolor{red}{\textless type(varying\_string):}\textgreater\ trackedOutflowAbundancesAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→)
Return a list of implementations that provide the given list of attributes for the trackedOutflowAbundances property of the hotHalo component.

tfamily \intzero trackedOutflowAbundancesCount() Compute the count of evolvable quantities in the trackedOutflowAbundances property of the HotHaloOutflowTracking component.

tfamily \textcolor{red}{\textless type(abundances):}\textgreater\ trackedOutflowAbundancesIsGettable()
Get the trackedOutflowAbundances property of the hotHalo component.

tfamily \logicalzero trackedOutflowAbundancesIsSettable() Specify whether the trackedOutflowAbundances property of the hotHalo component is settable.

tfamily \void trackedOutflowAbundancesRate(<type(abundances)> value) Cumulate to the rate of the trackedOutflowAbundances property of the HotHaloOutflowTracking component.

tfamily \void trackedOutflowAbundancesScale(<double> value) Set the scale of the trackedOutflowAbundances property of the HotHaloOutflowTracking component.

tfamily \void trackedOutflowAbundancesSet(<double> value) Set the trackedOutflowAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless double:}\textgreater\ trackedOutflowMass() Get the trackedOutflowMass property of the hotHalo component.

tfamily \textcolor{red}{\textless double:}\textgreater\ trackedOutflowMassIsGettable()
Get the trackedOutflowMass property of the hotHalo component.

tfamily \logicalzero trackedOutflowMassIsSettable() Specify whether the trackedOutflowMass property of the hotHalo component is settable.

tfamily \void trackedOutflowMassRate(<double> value) Cumulate to the rate of the trackedOutflowMass property of the HotHaloOutflowTracking component.

tfamily \void trackedOutflowMassScale(<double> value) Set the scale of the trackedOutflowMass property of the HotHaloOutflowTracking component.

tfamily \void trackedOutflowMassSet(<double> value) Set the trackedOutflowMass property of the hotHalo component.

tfamily \textcolor{red}{\textless double:}\textgreater\ unaccretedMass() Get the unaccretedMass property of the hotHalo component.
16.5. Objects

```plaintext
tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundances() Get the abundances property of the hotHalo component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesCold() Get the abundancesCold property of the hotHalo component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesColdIsGettable() Get the abundancesCold property of the hotHalo component.

tfamily \logicalzero abundancesColdIsSettable() Specify whether the abundancesCold property of the hotHalo component is settable.

tfamily \void abundancesColdRate(<type(abundances)> value) Cumulate to the rate of the abundancesCold property of the HotHaloColdMode component.

tfamily \void abundancesColdScale(<type(abundances)> value) Set the scale of the abundancesCold property of the HotHaloColdMode component.
```

501
tfamily \textcolor{red}{\textlesstype(abundances)} abundancesColdSet(<type(abundances)> value) Set the abundancesCold property of the hotHalo component.

tfamily \intzero abundancesCount() Compute the count of evolvable quantities in the abundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textlesstype(abundances)} abundancesIsGettable() Get the abundances property of the hotHalo component.

tfamily \logicalzero abundancesIsSettable() Specify whether the abundances property of the hotHalo component is settable.

tfamily \void abundancesRate(<type(abundances)> value) Cumulate to the rate of the abundances property of the HotHaloStandard component.

tfamily \void abundancesScale(<type(abundances)> value) Set the scale of the abundances property of the HotHaloStandard component.

tfamily \void abundancesSet(<type(abundances)> value) Set the abundances property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)} angularMomentum() Get the angularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)} angularMomentumAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the angularMomentum property of the hotHalo component.

tfamily \void angularMomentumCold(<double> value) Cumulate to the rate of the angularMomentumCold property of the HotHaloColdMode component.

tfamily \void angularMomentumColdScale(<double> value) Set the scale of the angularMomentumCold property of the HotHaloColdMode component.

tfamily \void angularMomentumColdSet(<double> value) Set the angularMomentumCold property of the hotHalo component.

502
tfamily \textcolor{red}{\textless\textbf{double}\textgreater } angularMomentumIsGettable() Get the angularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless\textbf{logical}\textgreater } angularMomentumIsSettable() Specify whether the angularMomentum property of the hotHalo component is settable.

tfamily \textcolor{red}{\textless\textbf{double}\textgreater } angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textbf{double}\textgreater } angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textbf{double}\textgreater } angularMomentumSet(<double> value) Set the angularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless\textbf{double}\textgreater } angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textbf{double}\textgreater } chemicals() Get the chemicals property of the hotHalo component.

tfamily \textcolor{red}{\textless\textbf{logical}\textgreater } chemicalsIsGettable() Get the chemicals property of the hotHalo component.

tfamily \textcolor{red}{\textless\textbf{logical}\textgreater } chemicalsIsSettable() Specify whether the chemicals property of the hotHalo component is settable.

tfamily \textcolor{red}{\textless\textbf{type(chemicalAbundances)\textgreater } chemicalsIsGettable() Get the chemicals property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textbf{type(chemicalAbundances)\textgreater } chemicalsIsSettable() Specify whether the chemicals property of the hotHalo component is settable.

tfamily \textcolor{red}{\textless\textbf{type(chemicalAbundances)\textgreater } chemicalsRate(<type(chemicalAbundances)> value) Cumulate to the rate of the chemicals property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textbf{type(chemicalAbundances)\textgreater } chemicalsScale(<type(chemicalAbundances)> value) Set the scale of the chemicals property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textbf{type(chemicalAbundances)\textgreater } chemicalsSet(<type(chemicalAbundances)> value) Set the chemicals property of the hotHalo component.

tfamily \textcolor{red}{\textless\textbf{logical}\textgreater } coldModeIsActive() Return whether the coldMode implementation of the hotHalo component class is active.

tfamily \textcolor{red}{\textless\textbf{function()}\textgreater } createFunctionSet(<function()> Set the function used to create HotHaloStandard components.

tfamily \double \textcolor{red}{\textless\textbf{double}(3)\textgreater } positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→ Compute the density.

tfamily \textcolor{red}{\textless\textbf{int}\textgreater } chemicalsCount() Compute the count of evolvable quantities in the chemicals property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textbf{int}\textgreater } chemicalsIsGettable() Get the chemicals property of the hotHalo component.

tfamily \textcolor{red}{\textless\textbf{logical}\textgreater } chemicalsIsSettable() Specify whether the chemicals property of the hotHalo component is settable.

tfamily \textcolor{red}{\textless\textbf{type(chemicalAbundances)\textgreater } chemicalsRate(<type(chemicalAbundances)> value) Cumulate to the rate of the chemicals property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textbf{type(chemicalAbundances)\textgreater } chemicalsScale(<type(chemicalAbundances)> value) Set the scale of the chemicals property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textbf{type(chemicalAbundances)\textgreater } chemicalsSet(<type(chemicalAbundances)> value) Set the chemicals property of the hotHalo component.

tfamily \void deserializeRates(\textless\text{double}\textgreater\ array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(\textless\text{double}\textgreater\ array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(\textless\text{double}\textgreater\ array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(\textless\text{integer}\textgreater\ fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \textcolor{red}{\textless\text{double}\textgreater} heatSource() Get the \textcolor{red}{heatSource} property of the \textcolor{red}{hotHalo} component.

tfamily \textcolor{red}{\textless\text{type(abundances)}\textgreater} hotHaloCoolingAbundances() Get the \textcolor{red}{hotHaloCoolingAbundances} property of the \textcolor{red}{hotHalo} component.

504
16.5. Objects

tfamily \textcolor{red}{\textless type(abundances)}, \textgreater{ hotHaloCoolingAbundancesIsGettable() Get the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \logicalzero hotHaloCoolingAbundancesIsSettable() Specify whether the hotHaloCoolingAbundances property of the hotHalo component is settable.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAbundancesRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \logicalzero hotHaloCoolingAbundancesRateIsAttached() Return whether the rate method of the hotHaloCoolingAbundances property of the hotHalo component has been attached to a function.

tfamily \textcolor{red}{\textless double\textgreater} hotHaloCoolingAngularMomentum() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying \_string):\textgreater} hotHaloCoolingAngularMomentumAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \intzero hotHaloCoolingAngularMomentumCount() Compute the count of evolvable quantities in the hotHaloCoolingAngularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} hotHaloCoolingAngularMomentumIsGettable() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \logicalzero hotHaloCoolingAngularMomentumIsSettable() Specify whether the hotHaloCoolingAngularMomentum property of the hotHalo component is settable.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \logicalzero hotHaloCoolingAngularMomentumRateIsAttached() Return whether the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component has been attached to a function.

tfamily \textcolor{red}{\textless double\textgreater} hotHaloCoolingMass() Get the hotHaloCoolingMass property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying \_string):\textgreater} hotHaloCoolingMassAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the hotHaloCoolingMass property of the hotHalo component.

tfamily \intzero hotHaloCoolingMassCount() Compute the count of evolvable quantities in the hotHaloCoolingMass property of the HotHaloStandard component.
16. Coding GALACTICUS

tfamily \textcolor{red}{\textless double\textgreater} hotHaloCoolingMassIsGettable() Get the \textit{hotHaloCoolingMass} property of the \textit{hotHalo} component.

tfamily \logicalzero hotHaloCoolingMassIsSettable() Specify whether the \textit{hotHaloCoolingMass} property of the \textit{hotHalo} component is settable.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the \textit{hotHaloCoolingMass} property of the \textit{hotHalo} component.

tfamily \void hotHaloCoolingMassRateFunction(<function()> deferredFunction) Set the function to be used for the \textit{rate} method of the \textit{hotHaloCoolingMass} property of the \textit{hotHalo} component.

tfamily \logicalzero hotHaloCoolingMassRateIsAttached() Return whether the rate method of the hotHaloCoolingMass property of the hotHalo component has been attached to a function.

tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}{\textless logical\textgreater} isInitialized() Get the isInitialized property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater isInitializedAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the isInitialized property of the hotHalo component.

tfamily \textcolor{red}{\textless logical\textgreater} isInitializedIsGettable() Get the isInitialized property of the hotHalo component.

tfamily \logicalzero isInitializedIsSettable() Specify whether the isInitialized property of the hotHalo component is settable.

tfamily \void isInitializedSet(<logical> value) Set the isInitialized property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater massAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the mass property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} mass() Get the mass property of the hotHalo component.

tfamily \void massSet(<logical> value) Set the mass property of the hotHalo component.

tfamily \intzero massColdCount() Compute the count of evolvable quantities in the massCold property of the HotHaloColdMode component.

tfamily \textcolor{red}{\textless double\textgreater} massColdIsGettable() Get the massCold property of the hotHalo component.

tfamily \logicalzero massColdIsSettable() Specify whether the massCold property of the hotHalo component is settable.
tfamily \void massColdRate(<double> value) Cumulate to the rate of the massCold property of the HotHaloColdMode component.

tfamily \void massColdScale(<double> value) Set the scale of the massCold property of the HotHaloColdMode component.

tfamily \void massColdSet(<double> value) Set the massCold property of the hotHalo component.

tfamily \int zero massCount() Compute the count of evolvable quantities in the mass property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} massIsGettable() Get the mass property of the hotHalo component.

tfamily \logical zero massIsSettable() Specify whether the mass property of the hotHalo component is settable.

tfamily \void massRate(<double> value) Cumulate to the rate of the mass property of the HotHaloStandard component.

tfamily \void massRemovalRate() Called whenever the standard hot halo component removes mass from the halo.

tfamily \void massScale(<double> value) Set the scale of the mass property of the HotHaloStandard component.

tfamily \void massSet(<double> value) Set the mass property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} massSink() Get the massSink property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater massSinkAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the massSink property of the hotHalo component.

tfamily \int zero massSinkCount() Compute the count of evolvable quantities in the massSink property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} massSinkIsGettable() Get the massSink property of the hotHalo component.

tfamily \logical zero massSinkIsSettable() Specify whether the massSink property of the hotHalo component is settable.

tfamily \void massSinkRate(<double> value) Cumulate to the rate of the massSink property of the hotHalo component.

tfamily \void massSinkRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the massSink property of the HotHaloStandard component.

tfamily \logical zero massSinkRateIsAttached() Return whether the rate method of the massSink property of the HotHaloStandard component has been attached to a function.

tfamily \void nameFromIndex(<integer> count →, <varying_string>name ←) Return the name of a property given is index.
16. Coding Galacticus

tfamily \logicalzero nullIsActive() Return whether the null implementation of the hotHalo component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \textcolor{red}{\textless double\textgreater} outerRadius() Get the outerRadius property of the hotHalo component.

tfamily \textcolor{red}{\textless textless type(varying\_string):(\textgreater}\textgreater} outerRadiusAttributeMatch(<logical>[requireGettable]:→,<logical>[requireSettable]→,<logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the outerRadius property of the hotHalo component.

tfamily \intzero outerRadiusCount() Compute the count of evolvable quantities in the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusFunction(<function()> deferredFunction) Set the function to be used for the get method of the outerRadius property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} outerRadiusGrowthRate() Get the outerRadiusGrowthRate property of the HotHaloColdMode component.

tfamily \void outerRadiusGrowthRateFunction(procedure(HotHaloColdModeouterRadiusGrowthRateInterface) deferredFunction) Set the function for the deferred outerRadiusGrowthRate property of the HotHaloColdMode component.

tfamily \logicalzero outerRadiusGrowthRateFunctionIsSet() Specify whether the deferred function for the outerRadiusGrowthRate property of the HotHaloColdMode component has been set.

tfamily \logicalzero outerRadiusIsAttached() Return whether the get method of the outerRadius property of the HotHaloStandard component has been attached to a function.

tfamily \textcolor{red}{\textless double\textgreater} outerRadiusIsGettable() Get the outerRadius property of the hotHalo component.

tfamily \logicalzero outerRadiusIsSettable() Specify whether the outerRadius property of the hotHalo component is settable.

tfamily \void outerRadiusRate(<double> value) Cumulate to the rate of the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusScale(<double> value) Set the scale of the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusSet(<double> value) Set the outerRadius property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} outerRadiusValue() Get the outerRadius property of the hotHalo component.

tfamily \textcolor{red}{\textless abundances\textgreater} outflowedAbundances() Get the outflowedAbundances property of the hotHalo component.
16.5. Objects

tfamily {textcolor{red}\textless double\textgreater} outflowedAbundances() Get the outflowedAbundances property of the hotHalo component.

tfamily int zero outflowedAbundancesCount() Compute the count of evolvable quantities in the outflowedAbundances property of the HotHaloStandard component.

tfamily {textcolor{red}\textless double\textgreater} outflowedAbundancesIsGettable() Get the outflowedAbundances property of the hotHalo component.

tfamily int zero outflowedAbundancesIsSettable() Specify whether the outflowedAbundances property of the hotHalo component is settable.

tfamily void outflowedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the outflowedAbundances property of the HotHaloStandard component.

tfamily void outflowedAbundancesScale(<type(abundances)> value) Set the scale of the outflowedAbundances property of the HotHaloStandard component.

tfamily void outflowedAbundancesSet(<type(abundances)> value) Set the outflowedAbundances property of the hotHalo component.

outflowedAbundancesAttributeMatch(<logical>[requireGettable]->, <logical>[requireSettable]->, <logical>[requireEvolvable]->) Return a list of implementations that provide the given list of attributes for the outflowedAbundances property of the hotHalo component.

tfamily {textless type(abundances):}\textgreater outflowedAbundancesIsGettable() Get the outflowedAbundances property of the hotHalo component.

outflowedAbundancesIsSettable() Specify whether the outflowedAbundances property of the hotHalo component is settable.

tfamily void outflowedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the outflowedAbundances property of the HotHaloStandard component.

tfamily void outflowedAbundancesScale(<type(abundances)> value) Set the scale of the outflowedAbundances property of the HotHaloStandard component.

tfamily void outflowedAbundancesSet(<type(abundances)> value) Set the outflowedAbundances property of the hotHalo component.

outflowedAbundancesAttributeMatch(<logical>[requireGettable]->, <logical>[requireSettable]->, <logical>[requireEvolvable]->) Return a list of implementations that provide the given list of attributes for the outflowedAbundances property of the hotHalo component.

outflowedAngularMomentum() Get the outflowedAngularMomentum property of the hotHalo component.

outflowedAngularMomentumAttributeMatch(<logical>[requireGettable]->, <logical>[requireSettable]->, <logical>[requireEvolvable]->) Return a list of implementations that provide the given list of attributes for the outflowedAngularMomentum property of the hotHalo component.

outflowedAngularMomentumCount() Compute the count of evolvable quantities in the outflowedAngularMomentum property of the HotHaloStandard component.

outflowedAngularMomentumIsGettable() Get the outflowedAngularMomentum property of the hotHalo component.

outflowedAngularMomentumIsSettable() Specify whether the outflowedAngularMomentum property of the hotHalo component is settable.

outflowedAngularMomentumRate(<double> value) Cumulate to the rate of the outflowedAngularMomentum property of the HotHaloStandard component.

outflowedAngularMomentumScale(<double> value) Set the scale of the outflowedAngularMomentum property of the HotHaloStandard component.

outflowedAngularMomentumSet(<double> value) Set the outflowedAngularMomentum property of the hotHalo component.

outflowedAngularMomentumAttributeMatch(<logical>[requireGettable]->, <logical>[requireSettable]->, <logical>[requireEvolvable]->) Return a list of implementations that provide the given list of attributes for the outflowedAngularMomentum property of the hotHalo component.

outflowedMass() Get the outflowedMass property of the hotHalo component.

outflowedMassAttributeMatch(<logical>[requireGettable]->, <logical>[requireSettable]->, <logical>[requireEvolvable]->) Return a list of implementations that provide the given list of attributes for the outflowedMass property of the hotHalo component.
16. Coding GALACTICUS

tfamily \intzero outflowedMassCount() Compute the count of evolvable quantities in the `outflowedMass` property of the `HotHaloStandard` component.

tfamily \textcolor{red}\{\textless double\textgreater\ outflowedMassIsGettable() Get the `outflowedMass` property of the `hotHalo` component.

tfamily \logicalzero outflowedMassIsSettable() Specify whether the `outflowedMass` property of the `hotHalo` component is settable.

tfamily \void outflowedMassRate(<double> value) Cumulate to the rate of the `outflowedMass` property of the `HotHaloStandard` component.

tfamily \void outflowedMassScale(<double> value) Set the scale of the `outflowedMass` property of the `HotHaloStandard` component.

tfamily \void outflowedMassSet(<double> value) Set the `outflowedMass` property of the `hotHalo` component.

tfamily \textcolor{red}\{\textless double\textgreater\ outflowingAbundances() Get the `outflowingAbundances` property of the `hotHalo` component.

tfamily \textcolor{red}\{\textless double\textgreater\ outflowingAbundancesIsGettable() Get the `outflowingAbundances` property of the `hotHalo` component.

tfamily \logicalzero outflowingAbundancesIsSettable() Specify whether the `outflowingAbundances` property of the `hotHalo` component is settable.

tfamily \void outflowingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the `outflowingAbundances` property of the `HotHaloStandard` component.

tfamily \void outflowingAbundancesRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the `outflowingAbundances` property of the `HotHaloStandard` component.

tfamily \logicalzero outflowingAbundancesRateIsAttached() Return whether the rate method of the `outflowingAbundances` property of the `HotHaloStandard` component has been attached to a function.

tfamily \textcolor{red}\{\textless double\textgreater\ outflowingAngularMomentum() Get the `outflowingAngularMomentum` property of the `hotHalo` component.

tfamily \textcolor{red}\{\textless double\textgreater\ outflowingAngularMomentumIsGettable() Get the `outflowingAngularMomentum` property of the `hotHalo` component.

tfamily \logicalzero outflowingAngularMomentumIsSettable() Specify whether the `outflowingAngularMomentum` property of the `hotHalo` component is settable.

tfamily \void outflowingAngularMomentumRate(<type(abundances)> value) Cumulate to the rate of the `outflowingAngularMomentum` property of the `HotHaloStandard` component.

tfamily \void outflowingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the `outflowingAngularMomentum` property of the `HotHaloStandard` component.

tfamily \intzero outflowingAngularMomentumCount() Compute the count of evolvable quantities in the `outflowingAngularMomentum` property of the `HotHaloStandard` component.
tfamily \textcolor{red}\{\textless double\textgreater\} outflowingAngularMomentumIsGettable() Get the *outflowingAngularMomentum* property of the *hotHalo* component.

tfamily \logicalzero outflowingAngularMomentumIsSettable() Specify whether the *outflowingAngularMomentum* property of the *hotHalo* component is settable.

tfamily \void outflowingAngularMomentumRate(<double> value) Cumulate to the rate of the *outflowingAngularMomentum* property of the *hotHalo* component.

tfamily \void outflowingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the *rate* method of the *outflowingAngularMomentum* property of the *HotHaloStandard* component.

tfamily \logicalzero outflowingAngularMomentumRateIsAttached() Return whether the rate method of the *outflowingAngularMomentum* property of the *HotHaloStandard* component has been attached to a function.

\textcolor{red}\{\textless double\textgreater\} outflowingMass() Get the *outflowingMass* property of the *hotHalo* component.

\textcolor{red}\{\textless double\textgreater\} outflowingMassAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the *outflowingMass* property of the *hotHalo* component.

tfamily \int zero outflowingMassCount() Compute the count of evolvable quantities in the *outflowingMass* property of the *HotHaloStandard* component.

tfamily \textcolor{red}\{\textless double\textgreater\} outflowingMassIsGettable() Get the *outflowingMass* property of the *hotHalo* component.

tfamily \logicalzero outflowingMassIsSettable() Specify whether the *outflowingMass* property of the *hotHalo* component is settable.

tfamily \void outflowingMassRate(<double> value) Cumulate to the rate of the *outflowingMass* property of the *hotHalo* component.

tfamily \void outflowingMassRateFunction(<function()> deferredFunction) Set the function to be used for the *rate* method of the *outflowingMass* property of the *HotHaloStandard* component.

tfamily \logicalzero outflowingMassRateIsAttached() Return whether the rate method of the *outflowingMass* property of the *HotHaloStandard* component has been attached to a function.

tfamily \textcolor{red}\{\textless void\textgreater\} outflowReturn() Get the *outflowReturn* property of the *HotHaloColdMode* component.

tfamily \void outflowReturnFunction(procedure(HotHaloColdModeoutflowReturnInterface) deferredFunction) Set the function for the deferred *outflowReturn* property of the *HotHaloColdMode* component.

tfamily \logicalzero outflowReturnFunctionIsSet() Specify whether the deferred function for the *outflowReturn* property of the *HotHaloColdMode* component has been set.

\logicalzero outflowTrackingIsActive() Return whether the outflowTracking implementation of the *hotHalo* component class is active.
16. Coding GALACTICUS

tfamily \void output(\text{integer}\text{Property}+, \text{integer} BufferCount+, \text{integer}(;:\text{Property}+) integerBuffer\leftrightarrow, \text{integer} doubleProperty\leftrightarrow, \text{integer} doubleBufferCount+, \text{double}(;:\text{Property}+) doubleBuffer\leftrightarrow, \text{double} time\rightarrow, \text{integer} instance\rightarrow) Generate values of outputtable properties.

tfamily \void outputCount(\text{integer} PropertyCount+, \text{integer} doublePropertyCount+, \text{double} time\rightarrow, \text{integer} instance\rightarrow) Compute a count of outputtable properties.

tfamily \void outputNames(\text{integer} Property+, \text{char} (:|) PropertyNames+, \text{char} (:|) PropertyComments+, \text{double}(;:\text{Property}+) doublePropertyNames+, \text{double}(;:\text{Property}+) doublePropertyComments+, \text{double}(;:\text{Property}+) doublePropertyUnitsSI+, \text{integer} doubleProperty\rightarrow, \text{double} time\rightarrow, \text{integer} instance\rightarrow) Generate names of outputtable properties.

tfamily \double zero potential(\text{double} radius\rightarrow, \text{componentType} [componentType]\rightarrow, \text{massType} [massType]\rightarrow, \text{logical} [haloLoaded]\rightarrow) Compute the gravitational potential.

tfamily \void readRaw(\text{integer} fileHandle\rightarrow) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \double zero rotationCurve(\text{double} radius\rightarrow, \text{componentType} [componentType]\rightarrow, \text{massType} [massType]\rightarrow, \text{logical} [haloLoaded]\rightarrow) Compute the rotation curve.

tfamily \double zero rotationCurveGradient(\text{double} radius\rightarrow, \text{componentType} [componentType]\rightarrow, \text{massType} [massType]\rightarrow, \text{logical} [haloLoaded]\rightarrow) Compute the rotation curve gradient.

tfamily \int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(\text{double}(:) array\rightarrow) Serialize the evolvable rates to an array.

tfamily \void serializeScales(\text{double}(:) array\rightarrow) Serialize the evolvable scales to an array.

tfamily \void serializeValues(\text{double}(:) array\rightarrow) Serialize the evolvable quantities to an array.

tfamily \logical zero standardIsActive() Return whether the standard implementation of the hotHalo component class is active.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} strippedAbundances() Get the strippedAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} strippedAbundancesAttributeMatch(\text{logical} [requireGettable]\rightarrow, \text{logical} [requireSettable]\rightarrow, \text{logical} [requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the strippedAbundances property of the hotHalo component.

tfamily \int zero strippedAbundancesCount() Compute the count of evolvable quantities in the strippedAbundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} strippedAbundancesIsGettable() Get the strippedAbundances property of the hotHalo component.

tfamily \logical zero strippedAbundancesIsSettable() Specify whether the strippedAbundances property of the hotHalo component is settable.

tfamily \void strippedAbundancesRate(\text{type(abundances)} value) Cumulate to the rate of the strippedAbundances property of the HotHaloStandard component.
16.5. Objects

tfamily\void strippedAbundancesScale(<type(abundances)> value) Set the scale of the strippedAbundances property of the HotHaloStandard component.

tfamily\void strippedAbundancesSet(<type(abundances)> value) Set the strippedAbundances property of the hotHalo component.

tfamily \textcolor{red}{{\textless double\textgreater}} strippedMass() Get the strippedMass property of the hotHalo component.

tfamily \textcolor{red}{{\textless type(varying\_string)(:)}\textgreater} strippedMassAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the strippedMass property of the hotHalo component

tfamily\intzero strippedMassCount() Compute the count of evolvable quantities in the strippedMass property of the HotHaloStandard component.

tfamily \textcolor{red}{{\textless type(abundances)}} trackedOutflowAbundances() Get the trackedOutflowAbundances property of the hotHalo component.

tfamily \textcolor{red}{{\textless type(varying\_string)(:)}\textgreater} trackedOutflowAbundancesAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the trackedOutflowAbundances property of the hotHalo component

tfamily\intzero trackedOutflowAbundancesCount() Compute the count of evolvable quantities in the trackedOutflowAbundances property of the HotHaloOutflowTracking component.

tfamily \textcolor{red}{{\textless type(abundances)}} trackedOutflowAbundancesIsGettable() Get the trackedOutflowAbundances property of the hotHalo component.

tfamily\logicalzero trackedOutflowAbundancesIsSettable() Specify whether the trackedOutflowAbundances property of the hotHalo component is settable.

tfamily\void trackedOutflowAbundancesRate(<type(abundances)> value) Cumulate to the rate of the trackedOutflowAbundances property of the HotHaloOutflowTracking component.
tfamily \ void trackedOutflowAbundancesScale(<type(abundances)> value) Set the scale of the trackedOutflowAbundances property of the HotHaloOutflowTracking component.

tfamily \ void trackedOutflowAbundancesSet(<type(abundances)> value) Set the trackedOutflowAbundances property of the hotHalo component.

 tfamily \ textcolor{red}{\textless\textgreater} trackedOutflowAbundancesSet(<type(abundances)> value) Set the trackedOutflowAbundances property of the hotHalo component.

tfamily \ textcolor{red}{\textlesstype(varying\_string)(:)} trackedOutflowMass() Get the trackedOutflowMass property of the hotHalo component.

tfamily \ textcolor{red}{\textlesstype(varying\_string)(:)} trackedOutflowMassAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the trackedOutflowMass property of the hotHalo component.

tfamily \ int zero trackedOutflowMassCount() Compute the count of evolvable quantities in the trackedOutflowMass property of the HotHaloOutflowTracking component.

 tfamily \ textcolor{red}{\textlesstype(varying\_string)(:)} trackedOutflowMassIsGettable() Get the trackedOutflowMass property of the hotHalo component.

 tfamily \ logical zero trackedOutflowMassIsSettable() Specify whether the trackedOutflowMass property of the hotHalo component is settable.

 tfamily \ void trackedOutflowMassRate(<double> value) Cumulate to the rate of the trackedOutflowMass property of the HotHaloOutflowTracking component.

tfamily \ void trackedOutflowMassScale(<double> value) Set the scale of the trackedOutflowMass property of the HotHaloOutflowTracking component.

 tfamily \ void trackedOutflowMassSet(<double> value) Set the trackedOutflowMass property of the hotHalo component.

 tfamily \ textcolor{red}{\textless\textgreater} trackedOutflowMassAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the trackedOutflowMass property of the hotHalo component.

 tfamily \ int zero trackedOutflowMassCount() Compute the count of evolvable quantities in the trackedOutflowMass property of the HotHaloOutflowTracking component.

 tfamily \ textcolor{red}{\textlesstype(varying\_string)(:)} trackedOutflowMassIsGettable() Get the trackedOutflowMass property of the hotHalo component.

 tfamily \ logical zero trackedOutflowMassIsSettable() Specify whether the trackedOutflowMass property of the hotHalo component is settable.

 tfamily \ void trackedOutflowMassRate(<double> value) Cumulate to the rate of the trackedOutflowMass property of the HotHaloOutflowTracking component.

 tfamily \ void trackedOutflowMassScale(<double> value) Set the scale of the trackedOutflowMass property of the hotHalo component.

 tfamily \ textcolor{red}{\textlesstype(varying\_string)(:)} type() Return the type of this object.

 tfamily \ textcolor{red}{\textlesstype(varying\_string)(:)} unaccretedMass() Get the unaccretedMass property of the hotHalo component.

 tfamily \ textcolor{red}{\textlesstype(varying\_string)(:)} unaccretedMassAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the unaccretedMass property of the hotHalo component.

 tfamily \ int zero unaccretedMassCount() Compute the count of evolvable quantities in the unaccretedMass property of the HotHaloStandard component.

 tfamily \ textcolor{red}{\textlesstype(varying\_string)(:)} unaccretedMassIsGettable() Get the unaccretedMass property of the hotHalo component.

 tfamily \ logical zero unaccretedMassIsSettable() Specify whether the unaccretedMass property of the hotHalo component is settable.

 tfamily \ void unaccretedMassRate(<double> value) Cumulate to the rate of the unaccretedMass property of the HotHaloStandard component.

 tfamily \ void unaccretedMassScale(<double> value) Set the scale of the unaccretedMass property of the HotHaloStandard component.
16.5. Objects

tfamily \ void unaccretedMassSet(<double> value) Set the unaccretedMass property of the hotHalo component.

tfamily \ logical zero verySimpleIsActive() Return whether the verySimple implementation of the hotHalo component class is active.

tfamily nodeComponentHotHaloNull

tfamily {\textcolor{red}{\textless type(abundances)\textgreater}} abundances() Get the abundances property of the hotHalo component.

tfamily {\textcolor{red}{\textless type(varying\_string)\textgreater}} abundancesAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the abundances property of the hotHalo component.

tfamily {\textcolor{red}{\textless type(abundances)\textgreater}} abundancesCold() Get the abundancesCold property of the hotHalo component.

tfamily {\textcolor{red}{\textless type(varying\_string)\textgreater}} abundancesColdAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the abundancesCold property of the hotHalo component.

tfamily \intzero abundancesColdCount() Compute the count of evolvable quantities in the abundancesCold property of the HotHaloColdMode component.

tfamily {\textcolor{red}{\textless type(abundances)\textgreater}} abundancesIsGettable() Get the abundancesCold property of the hotHalo component.

tfamily \ logical zero abundancesColdIsSettable() Specify whether the abundancesCold property of the hotHalo component is settable.

tfamily \ void abundancesColdRate(<type(abundances)> value) Cumulate to the rate of the abundancesCold property of the HotHaloColdMode component.

tfamily \ void abundancesColdScale(<type(abundances)> value) Set the scale of the abundancesCold property of the HotHaloColdMode component.

tfamily \ void abundancesColdSet(<type(abundances)> value) Set the abundancesCold property of the hotHalo component.

tfamily \intzero abundancesCount() Compute the count of evolvable quantities in the abundances property of the HotHaloStandard component.

tfamily {\textcolor{red}{\textless type(abundances)\textgreater}} abundancesIsGettable() Get the abundances property of the hotHalo component.

tfamily \ logical zero abundancesIsSettable() Specify whether the abundances property of the hotHalo component is settable.

tfamily \ void abundancesRate(<type(abundances)> value) Cumulate to the rate of the abundances property of the HotHaloStandard component.

tfamily \ void abundancesScale(<type(abundances)> value) Set the scale of the abundances property of the HotHaloStandard component.
tfamily \ void abundancesSet(<type(abundances)> value) Set the abundances property of the hotHalo component.

tfamily \ \textcolor{red}{\textless\textless\textgreater\textgreater\textless\textgreater} angularMomentum() Get the angularMomentum property of the hotHalo component.

tfamily \ \textcolor{red}{\textless\textless\textgreater\textless\textgreater} angularMomentumAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the angularMomentum property of the hotHalo component.

tfamily \ \textcolor{red}{\textless\textless\textgreater\textless\textgreater} angularMomentumCold() Get the angularMomentumCold property of the hotHalo component.

tfamily \ \textcolor{red}{\textless\textless\textgreater\textless\textgreater} angularMomentumColdAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the angularMomentumCold property of the hotHalo component.

tfamily \ \textcolor{red}{\textless\textless\textgreater\textless\textgreater} angularMomentumColdCount() Compute the count of evolvable quantities in the angularMomentumCold property of the hotHaloColdMode component.

tfamily \ \textcolor{red}{\textless\textless\textgreater\textless\textgreater} angularMomentumColdIsGettable() Get the angularMomentumCold property of the hotHalo component.

tfamily \ \textcolor{red}{\textless\textless\textgreater\textless\textgreater} angularMomentumColdIsSettable() Specify whether the angularMomentumCold property of the hotHalo component is settable.

tfamily \ void angularMomentumColdRate(<double> value) Cumulate to the rate of the angularMomentumCold property of the hotHaloColdMode component.

tfamily \ void angularMomentumColdScale(<double> value) Set the scale of the angularMomentumCold property of the hotHaloColdMode component.

tfamily \ void angularMomentumColdSet(<double> value) Set the angularMomentumCold property of the hotHalo component.

tfamily \ \intzero angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the hotHaloStandard component.

tfamily \ \textcolor{red}{\textless\textless\textgreater\textless\textgreater} angularMomentumIsGettable() Get the angularMomentum property of the hotHalo component.

tfamily \ \textcolor{red}{\textless\textless\textgreater\textless\textgreater} angularMomentumIsSettable() Specify whether the angularMomentum property of the hotHalo component is settable.

tfamily \ void angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the hotHaloStandard component.

tfamily \ void angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the hotHaloStandard component.

tfamily \ void angularMomentumSet(<double> value) Set the angularMomentum property of the hotHalo component.
16.5. Objects

tfamily \void builder(\*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily \textcolor{red}{\textless type(chemicalAbundances)>\textgreater} chemicals() Get the chemicals property of the hotHalo component.

tfamily \textcolor{red}{\textless varying\_string()>\textgreater} chemicalsAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the chemicals property of the hotHalo component.

tfamily \intzero chemicalsCount() Compute the count of evolvable quantities in the chemicals property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless type(chemicalAbundances)>\textgreater} chemicalsIsGettable() Get the chemicals property of the hotHalo component.

tfamily \logicalzero chemicalsIsSettable() Specify whether the chemicals property of the hotHalo component is settable.

tfamily \void chemicalsRate(<type(chemicalAbundances)> value) Cumulate to the rate of the chemicals property of the HotHaloStandard component.

tfamily \void chemicalsScale(<type(chemicalAbundances)> value) Set the scale of the chemicals property of the HotHaloStandard component.

tfamily \void chemicalsSet(<type(chemicalAbundances)> value) Set the chemicals property of the hotHalo component.

tfamily \logicalzero coldModeIsActive() Return whether the coldMode implementation of the hotHalo component class is active.

tfamily \doublezero density(<double(3)> positionSpherical→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless double\textgreater} heatSource() Get the heatSource property of the hotHalo component.
tfamily \textcolor{red}{\textless double\textgreater} heatSourceRate(<double> value) Cumulate to the rate of the heatSource property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} hotHaloCoolingAngularMomentum() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} hotHaloCoolingAngularMomentumAttributeMatch(<logical>[requireGettable]->, <logical>[requireSettable]->, <logical>[requireEvolvable]->) Return a list of implementations that provide the given list of attributes for the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} hotHaloCoolingAngularMomentumIsAttached() Return whether the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component has been attached to a function.

tfamily \textcolor{red}{\textless type(varying\_string)(:)} hotHaloCoolingAngularMomentumAt- ttributeMatch(<logical>[requireGettable]->, <logical>[requireSettable]->, <logical>[requireEvolvable]->) Return a list of implementations that provide the given list of attributes for the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)} hotHaloCoolingAngularMomentumAt- ttributeMatch(<logical>[requireGettable]->, <logical>[requireSettable]->, <logical>[requireEvolvable]->) Return a list of implementations that provide the given list of attributes for the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)} hotHaloCoolingAngularMomentumAt- ttributeMatch(<logical>[requireGettable]->, <logical>[requireSettable]->, <logical>[requireEvolvable]->) Return a list of implementations that provide the given list of attributes for the hotHaloCoolingAngularMomentum property of the hotHalo component.

518
16.5. Objects

tfamily $\intzero \text{hotHaloCoolingAngularMomentumCount()}$ Compute the count of evolvable quantities in the hotHaloCoolingAngularMomentum property of the HotHaloStandard component.

tfamily $\textcolor{red}{\textless \text{double}\textgreater} \text{hotHaloCoolingAngularMomentumIsGettable()}$ Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily $\logicalzero \text{hotHaloCoolingAngularMomentumIsSettable()}$ Specify whether the hotHaloCoolingAngularMomentum property of the hotHalo component is settable.

tfamily $\void \text{hotHaloCoolingAngularMomentumRate(<double> value)}$ Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily $\void \text{hotHaloCoolingAngularMomentumRateFunction(<function()> deferredFunction)}$ Set the function to be used for the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily $\logicalzero \text{hotHaloCoolingAngularMomentumRateIsAttached()}$ Return whether the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component has been attached to a function.

tfamily $\textcolor{red}{\textless \text{double}\textgreater} \text{hotHaloCoolingMass()}$ Get the hotHaloCoolingMass property of the hotHalo component.

tfamily $\textcolor{red}{\textless \text{type(varying\_string)(:)}\textgreater} \text{hotHaloCoolingMassAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→)}$ Return a list of implementations that provide the given list of attributes for the hotHaloCoolingMass property of the hotHalo component.

tfamily $\intzero \text{hotHaloCoolingMassCount()}$ Compute the count of evolvable quantities in the hotHaloCoolingMass property of the HotHaloStandard component.

tfamily $\textcolor{red}{\textless \text{double}\textgreater} \text{hotHaloCoolingMassIsGettable()}$ Get the hotHaloCoolingMass property of the hotHalo component.

tfamily $\logicalzero \text{hotHaloCoolingMassIsSettable()}$ Specify whether the hotHaloCoolingMass property of the hotHalo component is settable.

tfamily $\void \text{hotHaloCoolingMassRate(<double> value)}$ Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily $\void \text{hotHaloCoolingMassRateFunction(<function()> deferredFunction)}$ Set the function to be used for the rate method of the hotHaloCoolingMass property of the hotHalo component.

tfamily $\logicalzero \text{hotHaloCoolingMassRateIsAttached()}$ Return whether the rate method of the hotHaloCoolingMass property of the hotHalo component has been attached to a function.

tfamily $\void \text{initialize()}$ Initialize the object.

tfamily $\textcolor{red}{\textless \text{logical}\textgreater} \text{isInitialized()}$ Get the isInitialized property of the hotHalo component.

tfamily $\textcolor{red}{\textless \text{type(varying\_string)(:)}\textgreater} \text{isInitializedAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→)}$ Return a list of implementations that provide the given list of attributes for the isInitialized property of the hotHalo component.
tfamily \textcolor{red}{\textless \text{logical}\textgreater} \text{isInitializedIsGettable}() Get the \texttt{isInitialized} property of the hotHalo component.

tfamily \texttt{logicalzero} \text{isInitializedIsSettable}() Specify whether the \texttt{isInitialized} property of the hotHalo component is settable.

tfamily \texttt{void} \text{isInitializedSet}(<\text{logical}> \text{value}) Set the \texttt{isInitialized} property of the hotHalo component.

tfamily \textcolor{red}{\textless \text{double}\textgreater} \text{mass}() Get the \texttt{mass} property of the hotHalo component.

tfamily \textcolor{red}{\textless \text{double}\textgreater} \text{massCold}() Get the \texttt{massCold} property of the hotHalo component.

tfamily \text{massColdAttributeMatch}(<\text{logical}>[\text{requireGettable}] \rightarrow, <\text{logical}>[\text{requireSettable}] \rightarrow, <\text{logical}>[\text{requireEvolvable}] \rightarrow) Return a list of implementations that provide the given list off attributes for the \texttt{mass} property of the hotHalo component.

tfamily \text{massColdCount}() Compute the count of evolvable quantities in the \texttt{massCold} property of the HotHaloColdMode component.

tfamily \text{massColdGettable}() Get the \texttt{massCold} property of the hotHalo component.

tfamily \text{massColdIsSettable}() Specify whether the \texttt{massCold} property of the hotHalo component is settable.

tfamily \text{massColdRate}(<\text{double}> \text{value}) Cumulate to the rate of the \texttt{massCold} property of the HotHaloColdMode component.

tfamily \text{massColdScale}(<\text{double}> \text{value}) Set the scale of the \texttt{massCold} property of the HotHaloColdMode component.

tfamily \text{massColdSet}(<\text{double}> \text{value}) Set the \texttt{massCold} property of the hotHalo component.

tfamily \text{massCount}() Compute the count of evolvable quantities in the \texttt{mass} property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless \text{double}\textgreater} \text{massGettable}() Get the \texttt{mass} property of the hotHalo component.

tfamily \text{massIsSettable}() Specify whether the \texttt{mass} property of the hotHalo component is settable.

tfamily \text{massRate}(<\text{double}> \text{value}) Cumulate to the rate of the \texttt{mass} property of the HotHaloStandard component.

tfamily \text{massScale}(<\text{double}> \text{value}) Set the scale of the \texttt{mass} property of the HotHaloStandard component.

tfamily \text{massSet}(<\text{double}> \text{value}) Set the \texttt{mass} property of the hotHalo component.
16.5. Objects

tfamily \textcolor{red}{\textless double\textgreater} massSink() Get the massSink property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} massSinkAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the massSink property of the hotHalo component.

tfamily \intzero massSinkCount() Compute the count of evolvable quantities in the massSink property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} massSinkIsGettable() Get the massSink property of the hotHalo component.

tfamily \logicalzero massSinkIsSettable() Specify whether the massSink property of the hotHalo component is settable.

tfamily \void massSinkRate(<double> value) Cumulate to the rate of the massSink property of the HotHaloStandard component.

tfamily \void nameFromIndex(<integer> count \rightarrow, <varying_string>name \leftarrow) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the hotHalo component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \textcolor{red}{\textless double\textgreater} outerRadius() Get the outerRadius property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} outerRadiusAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the outerRadius property of the hotHalo component.

tfamily \intzero outerRadiusCount() Compute the count of evolvable quantities in the outerRadius property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} outerRadiusIsGettable() Get the outerRadius property of the hotHalo component.

tfamily \logicalzero outerRadiusIsSettable() Specify whether the outerRadius property of the hotHalo component is settable.

tfamily \void outerRadiusRate(<double> value) Cumulate to the rate of the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusScale(<double> value) Set the scale of the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusSet(<double> value) Set the outerRadius property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype(abundances)\textgreater} outflowedAbundances() Get the outflowedAbundances property of the hotHalo component.
16. Coding GALACTICUS

tfamily \textcolor{red}{\textless} double \textgreater{} outflowedAngularMomentum() Get the \textit{outflowedAngularMomentum} property of the \textit{hotHalo} component.

tfamily \textcolor{red}{\textless} double \textgreater{} outflowedAngularMomentumIsGettable() Get the \textit{outflowedAngularMomentum} property of the \textit{hotHalo} component.

tfamily \logicalzero outflowedAngularMomentumIsSettable() Specify whether the \textit{outflowedAngularMomentum} property of the \textit{hotHalo} component is settable.

tfamily \void outflowedAngularMomentumRate(<double> value) Cumulate to the rate of the \textit{outflowedAngularMomentum} property of the \textit{HotHaloStandard} component.

tfamily \void outflowedAngularMomentumScale(<double> value) Set the scale of the \textit{outflowedAngularMomentum} property of the \textit{HotHaloStandard} component.

tfamily \void outflowedAngularMomentumSet(<double> value) Set the \textit{outflowedAngularMomentum} property of the \textit{hotHalo} component.

tfamily \textcolor{red}{\textless} double \textgreater{} outflowedMass() Get the \textit{outflowedMass} property of the \textit{hotHalo} component.

tfamily \textcolor{red}{\textless} double \textgreater{} outflowedMassIsGettable() Get the \textit{outflowedMass} property of the \textit{hotHalo} component.

tfamily \logicalzero outflowedMassIsSettable() Specify whether the \textit{outflowedMass} property of the \textit{hotHalo} component is settable.

tfamily \void outflowedMassRate(<double> value) Cumulate to the rate of the \textit{outflowedMass} property of the \textit{HotHaloStandard} component.

tfamily \void outflowedMassScale(<double> value) Set the scale of the \textit{outflowedMass} property of the \textit{HotHaloStandard} component.

tfamily \void outflowedMassSet(<double> value) Set the \textit{outflowedMass} property of the \textit{hotHalo} component.

522
16.5. Objects

- `intzero outflowedMassCount()` Compute the count of evolvable quantities in the `outflowedMass` property of the `HotHaloStandard` component.

- `textcolor{red}{\textless double\textgreater} outflowedMassIsGettable()` Get the `outflowedMass` property of the `hotHalo` component.

- `logicalzero outflowedMassIsSettable()` Specify whether the `outflowedMass` property of the `hotHalo` component is settable.

- `void outflowedMassRate(<double> value)` Cumulate to the rate of the `outflowedMass` property of the `HotHaloStandard` component.

- `void outflowedMassScale(<double> value)` Set the scale of the `outflowedMass` property of the `HotHaloStandard` component.

- `void outflowedMassSet(<double> value)` Set the `outflowedMass` property of the `hotHalo` component.

- `textcolor{red}{\textless double\textgreater} outflowingAbundances()` Get the `outflowingAbundances` property of the `hotHalo` component.

- `textcolor{red}{\textlesstype(varying\_string)(:)\textgreater} outflowingAbundancesAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→)` Return a list of implementations that provide the given list of attributes for the `outflowingAbundances` property of the `hotHalo` component.

- `intzero outflowingAbundancesCount()` Compute the count of evolvable quantities in the `outflowingAbundances` property of the `HotHaloStandard` component.

- `textcolor{red}{\textless double\textgreater} outflowingAbundancesIsGettable()` Get the `outflowingAbundances` property of the `hotHalo` component.

- `logicalzero outflowingAbundancesIsSettable()` Specify whether the `outflowingAbundances` property of the `hotHalo` component is settable.

- `void outflowingAbundancesRate(<type(abundances)> value)` Cumulate to the rate of the `outflowingAbundances` property of the `HotHaloStandard` component.

- `textcolor{red}{\textless double\textgreater} outflowingAngularMomentum()` Get the `outflowingAngularMomentum` property of the `hotHalo` component.

- `textcolor{red}{\textlesstype(varying\_string)(:)\textgreater} outflowingAngularMomentumAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→)` Return a list of implementations that provide the given list of attributes for the `outflowingAngularMomentum` property of the `hotHalo` component.

- `intzero outflowingAngularMomentumCount()` Compute the count of evolvable quantities in the `outflowingAngularMomentum` property of the `HotHaloStandard` component.

- `textcolor{red}{\textless double\textgreater} outflowingAngularMomentumIsGettable()` Get the `outflowingAngularMomentum` property of the `hotHalo` component.

- `logicalzero outflowingAngularMomentumIsSettable()` Specify whether the `outflowingAngularMomentum` property of the `hotHalo` component is settable.
Coding GALACTICUS

16. Coding GALACTICUS

tfamily \void outflowingAngularMomentumRate(<double> value) Cumulate to the rate of the outflowingAngularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} outflowingMass() Get the outflowingMass property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater outflowingMassAttributeMatch(<logical>[requireGettable] \rightarrow \textfamily{<logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow}) Return a list of implementations that provide the given list off attributes for the outflowingMass property of the hotHalo component.

tfamily \intzero outflowingMassCount() Compute the count of evolvable quantities in the outflowingMass property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater outflowingMassIsGettable() Get the outflowingMass property of the hotHalo component.

tfamily \logical zero outflowingMassIsSettable() Specify whether the outflowingMass property of the hotHalo component is settable.

tfamily \void outflowingMassRate(<double> value) Cumulate to the rate of the outflowingMass property of the HotHaloStandard component.

tfamily \logical zero outflowTrackingIsActive() Return whether the outflowTracking implementation of the hotHalo component class is active.

tfamily \void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer>[;;:] integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double[;;:]> doubleBuffer ↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty ↔, <char[*]:] integerPropertyNames ↔, <char[*]:] integerPropertyComments ↔, <double[;:]> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char[*]:] doublePropertyNames ↔, <char[*]:] doublePropertyComments ↔, <double[;:]> doublePropertyUnitsSI ↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \double zero potential(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

tfamily \void readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \double zero rotationCurve(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

tfamily \double zero rotationCurveGradient(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double[;:]:] array) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double[;:]:] array) Serialize the evolvable scales to an array.
16.5. Objects

tfamily void serializeValues(<double()> array) Serialize the evolvable quantities to an array.

tfamily logical zero standardIsActive() Return whether the standard implementation of the hotHalo component class is active.

tfamily textcolor{red}{textless type(abundances)\textgreater} strippedAbundances() Get the strippedAbundances property of the hotHalo component.

tfamily textcolor{red}{textless type(varying_string)\textgreater} strippedAbundancesAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list off attributes for the strippedAbundances property of the hotHalo component.

tfamily int zero strippedAbundancesCount() Compute the count of evolvable quantities in the strippedAbundances property of the HotHaloStandard component.

tfamily textcolor{red}{textless type(abundances)\textgreater} strippedAbundancesIsGettable() Get the strippedAbundances property of the hotHalo component.

tfamily logical zero strippedAbundancesIsSettable() Specify whether the strippedAbundances property of the hotHalo component is settable.

tfamily void strippedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the strippedAbundances property of the HotHaloStandard component.

tfamily void strippedAbundancesScale(<type(abundances)> value) Set the scale of the strippedAbundances property of the HotHaloStandard component.

tfamily void strippedAbundancesSet(<type(abundances)> value) Set the strippedAbundances property of the hotHalo component.

tfamily textcolor{red}{textless double\textgreater} strippedMass() Get the strippedMass property of the hotHalo component.

tfamily textcolor{red}{textless type(varying_string)\textgreater} strippedMassAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list off attributes for the strippedMass property of the hotHalo component.

tfamily int zero strippedMassCount() Compute the count of evolvable quantities in the strippedMass property of the HotHaloStandard component.

tfamily textcolor{red}{textless double\textgreater} strippedMassIsGettable() Get the strippedMass property of the hotHalo component.

tfamily logical zero strippedMassIsSettable() Specify whether the strippedMass property of the hotHalo component is settable.

tfamily void strippedMassRate(<double> value) Cumulate to the rate of the strippedMass property of the HotHaloStandard component.

tfamily void strippedMassScale(<double> value) Set the scale of the strippedMass property of the HotHaloStandard component.

tfamily void strippedMassSet(<double> value) Set the strippedMass property of the hotHalo component.
tfamily \textcolor{red}{\textless type(abundances)\textgreater} trackedOutflowAbundances() Get the trackedOutflowAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} trackedOutflowAbundancesAttributeMatch(\textcolor{red}{\textless logical\[requireGettable\]}\rightarrow, \textcolor{red}{\textless logical\[requireSettable\]}\rightarrow, \textcolor{red}{\textless logical\[requireEvolvable\]}\rightarrow) Return a list of implementations that provide the given list of attributes for the trackedOutflowAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype(abundances)\textgreater} trackedOutflowAbundancesIsGettable() Get the trackedOutflowAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} trackedOutflowMass() Get the trackedOutflowMass property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)\textgreater} trackedOutflowMassAttributeMatch(\textcolor{red}{\textless logical\[requireGettable\]}\rightarrow, \textcolor{red}{\textless logical\[requireSettable\]}\rightarrow, \textcolor{red}{\textless logical\[requireEvolvable\]}\rightarrow) Return a list of implementations that provide the given list of attributes for the trackedOutflowMass property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype(abundances)\textgreater} trackedOutflowMassIsGettable() Get the trackedOutflowMass property of the hotHalo component.

16. Coding GALACTICUS
16.5. Objects

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

tfamily \textcolor{red}{\textless double\textgreater} unaccretedMass() Get the unaccretedMass property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} unaccretedMassAttributeMatch(<logical>[requireGettable], <logical>[requireSettable], <logical>[requireEvolvable]) Return a list of implementations that provide the given list of attributes for the unaccretedMass property of the hotHalo component.

tfamily \intzerounaccretedMassCount() Compute the count of evolvable quantities in the unaccretedMass property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} unaccretedMassIsGettable() Get the unaccretedMass property of the hotHalo component.

tfamily \logicalzero unaccretedMassIsSettable() Specify whether the unaccretedMass property of the hotHalo component is settable.

tfamily \void unaccretedMassRate(<double> value) Cumulate to the rate of the unaccretedMass property of the HotHaloStandard component.

tfamily \void unaccretedMassScale(<double> value) Set the scale of the unaccretedMass property of the HotHaloStandard component.

tfamily \void unaccretedMassSet(<double> value) Set the unaccretedMass property of the hotHalo component.

tfamily \logicalzero verySimpleIsActive() Return whether the verySimple implementation of the hotHalo component class is active.

tfamily nodeComponentHotHaloOutflowTracking

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundances() Get the abundances property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} abundancesAttributeMatch(<logical>[requireGettable], <logical>[requireSettable], <logical>[requireEvolvable]) Return a list of implementations that provide the given list of attributes for the abundances property of the hotHalo component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesCold() Get the abundancesCold property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} abundancesColdAttributeMatch(<logical>[requireGettable], <logical>[requireSettable], <logical>[requireEvolvable]) Return a list of implementations that provide the given list of attributes for the abundancesCold property of the hotHalo component.

tfamily \intzero abundancesColdCount() Compute the count of evolvable quantities in the abundancesCold property of the HotHaloColdMode component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesColdIsGettable() Get the abundancesCold property of the hotHalo component.

tfamily \logicalzero abundancesColdIsSettable() Specify whether the abundancesCold property of the hotHalo component is settable.
16. Coding GALACTICUS

tfamily \void abundancesColdRate(<type(abundances)> value) Cumulate to the rate of the abundancesCold property of the HotHaloColdMode component.

tfamily \void abundancesColdScale(<type(abundances)> value) Set the scale of the abundancesCold property of the HotHaloColdMode component.

tfamily \void abundancesColdSet(<type(abundances)> value) Set the abundancesCold property of the hotHalo component.

tfamily \intzero abundancesCount() Compute the count of evolvable quantities in the abundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentum() Get the angularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumCold() Get the angularMomentumCold property of the hotHalo component.

528
16.5. Objects

tfamily\void\angularMomentumColdScale(<double> value) Set the scale of the \textit{angularMomentumCold} property of the \texttt{HotHaloColdMode} component.

tfamily\void\angularMomentumColdSet(<double> value) Set the \textit{angularMomentumCold} property of the \texttt{hotHalo} component.

tfamily\intzero\angularMomentumCount() Compute the count of evolvable quantities in the \textit{angularMomentum} property of the \texttt{HotHaloStandard} component.

tfamily\textcolor{red}{\textless double\textgreater}\angularMomentumIsGettable() Get the \textit{angularMomentum} property of the \texttt{hotHalo} component.

tfamily\logicalzero\angularMomentumIsSettable() Specify whether the \textit{angularMomentum} property of the \texttt{hotHalo} component is settable.

tfamily\void\angularMomentumRate(<double> value) Cumulate to the rate of the \textit{angularMomentum} property of the \texttt{HotHaloStandard} component.

tfamily\void\angularMomentumScale(<double> value) Set the scale of the \textit{angularMomentum} property of the \texttt{HotHaloStandard} component.

tfamily\void\angularMomentumSet(<double> value) Set the \textit{angularMomentum} property of the \texttt{hotHalo} component.

tfamily\void\builder(<\*type(node)>componentDefinition→) Build a \texttt{nodeComponent} from a supplied XML definition.

tfamily\textcolor{red}{\textless\textless type(chemicalAbundances)\textless\textgreater}\chemicals() Get the \textit{chemicals} property of the \texttt{hotHalo} component.

tfamily\textcolor{red}{\textless\textless type(varying\_string)::\textless\textgreater}\chemicalsAttributeMatch(<logical>|requireGettable|→, <logical>|requireSettable|→, <logical>|requireEvolvable|→) Return a list of implementations that provide the given list of attributes for the \textit{chemicals} property of the \texttt{hotHalo} component.

tfamily\intzero\chemicalsCount() Compute the count of evolvable quantities in the \textit{chemicals} property of the \texttt{HotHaloStandard} component.

tfamily\textcolor{red}{\textless\textless type(chemicalAbundances)\textless\textgreater}\chemicalsIsGettable() Get the \textit{chemicals} property of the \texttt{hotHalo} component.

tfamily\logicalzero\chemicalsIsSettable() Specify whether the \textit{chemicals} property of the \texttt{hotHalo} component is settable.

tfamily\void\chemicalsRate(<\texttt{chemicalAbundances}> value) Cumulate to the rate of the \textit{chemicals} property of the \texttt{HotHaloStandard} component.

tfamily\void\chemicalsScale(<\texttt{chemicalAbundances}> value) Set the scale of the \textit{chemicals} property of the \texttt{HotHaloStandard} component.

tfamily\void\chemicalsSet(<\texttt{chemicalAbundances}> value) Set the \textit{chemicals} property of the \texttt{hotHalo} component.

tfamily\logicalzero\coldModeIsActive() Return whether the \textit{coldMode} implementation of the \texttt{hotHalo} component class is active.

529
tfamily \void createFunctionSet(\langle function() \rangle) Set the function used to create \texttt{HotHaloStandard} components.

tfamily \doublezero density(\langle double(3) \rangle positionSpherical \rightarrow, \langle componentType \rangle [componentType] \rightarrow, \langle massType \rangle [massType] \rightarrow, \langle weightBy \rangle [weightBy] \rightarrow, \langle integer \rangle [weightIndex] \rightarrow, \langle logical \rangle [haloLoaded] \rightarrow) Compute the density.

tfamily \void deserializeRates(\langle double() \rangle array \leftarrow) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(\langle double() \rangle array \leftarrow) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(\langle double() \rangle array \leftarrow) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(\langle integer \rangle fileHandle \rightarrow) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(\langle double \rangle radius \rightarrow, \langle componentType \rangle [componentType] \rightarrow, \langle massType \rangle [massType] \rightarrow, \langle weightBy \rangle [weightBy] \rightarrow, \langle integer \rangle [weightIndex] \rightarrow, \langle logical \rangle [haloLoaded] \rightarrow) Compute the mass enclosed within a radius.

tfamily \red \textless double \langle heatSource() \rangle Get the \texttt{heatSource} property of the \texttt{hotHalo} component.

tfamily \red \textless type(varying\_string)(:): \langle heatSourceAttributeMatch(\langle logical \rangle [requireGettable] \rightarrow, \langle logical \rangle [requireSettable] \rightarrow, \langle logical \rangle [requireEvolvable] \rightarrow) \rangle \rangle Return a list of implementations that provide the given list off attributes for the \texttt{heatSource} property of the \texttt{hotHalo} component.

tfamily \intzero heatSourceCount() Compute the count of evolvable quantities in the \texttt{heatSource} property of the \texttt{HotHaloStandard} component.

tfamily \red \textless double \langle heatSourceIsGettable() \rangle Get the \texttt{heatSource} property of the \texttt{hotHalo} component.

tfamily \logicalzero heatSourceIsSettable() Specify whether the \texttt{heatSource} property of the \texttt{hotHalo} component is settable.

tfamily \void heatSourceRate(\langle double \rangle value) Cumulate to the rate of the \texttt{heatSource} property of the \texttt{hotHalo} component.

tfamily \void heatSourceRateFunction(\langle function() \rangle deferredFunction) Set the function to be used for the rate method of the \texttt{heatSource} property of the \texttt{HotHaloStandard} component.

tfamily \logicalzero heatSourceRateIsAttached() Return whether the rate method of the heatSource property of the \texttt{HotHaloStandard} component has been attached to a function.

tfamily \red \textless *type(treeNode) \langle host() \rangle Return a pointer to the host \texttt{treeNode} object.

tfamily \red \textless type(abundances) \langle hotHaloCoolingAbundances() \rangle Get the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component.
16.5. Objects

tfamily {textcolor{red}\{\textless\ type(varying\_string):\}\textgreater} hotHaloCoolingAbundancesAttrib-


tReturn a list of implementations that provide the given list of attributes for the hotHaloCoolingAbundances property of the hotHalo component.

tfamily {intzero\} hotHaloCoolingAbundancesCount() Compute the count of evolvable quantities in the hotHaloCoolingAbundances property of the HotHaloStandard component.

tfamily {textcolor{red}\{\textless\ type(abundances):\}\textgreater} hotHaloCoolingAbundancesIsGettable() Get the hotHaloCoolingAbundances property of the hotHalo component.

tfamily {logicalzero\} hotHaloCoolingAbundancesIsSettable() Specify whether the hotHaloCoolingAbundances property of the hotHalo component is settable.

tfamily {void\} hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily {void\} hotHaloCoolingAbundancesRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily {logicalzero\} hotHaloCoolingAbundancesRateIsAttached() Return whether the rate method of the hotHaloCoolingAbundances property of the hotHalo component has been attached to a function.

tfamily {textcolor{red}\{\textless\ double\}\textgreater} hotHaloCoolingAngularMomentum() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily {textcolor{red}\{\textless\ type(varying\_string):\}\textgreater} hotHaloCoolingAngularMomentumAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily {intzero\} hotHaloCoolingAngularMomentumCount() Compute the count of evolvable quantities in the hotHaloCoolingAngularMomentum property of the HotHaloStandard component.

tfamily {textcolor{red}\{\textless\ double\}\textgreater} hotHaloCoolingAngularMomentumIsGettable() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily {logicalzero\} hotHaloCoolingAngularMomentumIsSettable() Specify whether the hotHaloCoolingAngularMomentum property of the hotHalo component is settable.

tfamily {void\} hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily {void\} hotHaloCoolingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily {logicalzero\} hotHaloCoolingAngularMomentumRateIsAttached() Return whether the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component has been attached to a function.

tfamily {textcolor{red}\{\textless\ double\}\textgreater} hotHaloCoolingMass() Get the hotHaloCoolingMass property of the hotHalo component.
16. Coding GALACTICUS

tfamily \textcolor{red}{\textless\textgreater} hotHaloCoolingMassAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list off attributes for the hotHaloCoolingMass property of the hotHalo component.

tfamily \intzero hotHaloCoolingMassCount() Compute the count of evolvable quantities in the hotHaloCoolingMass property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textgreater} hotHaloCoolingMassIsGettable() Get the hotHaloCoolingMass property of the hotHalo component.

tfamily \logicalzero hotHaloCoolingMassIsSettable() Specify whether the hotHaloCoolingMass property of the hotHalo component is settable.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void hotHaloCoolingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingMass property of the hotHalo component.

tfamily \logicalzero hotHaloCoolingMassRateIsAttached() Return whether the rate method of the hotHaloCoolingMass property of the hotHalo component has been attached to a function.

tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}{\textless\textgreater} isInitialized() Get the isInitialized property of the hotHalo component.

tfamily \textcolor{red}{\textless\textgreater} isInitializedAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list off attributes for the isInitialized property of the hotHalo component.

tfamily \textcolor{red}{\textless\textgreater} isInitializedIsGettable() Get the isInitialized property of the hotHalo component.

tfamily \logicalzero isInitializedIsSettable() Specify whether the isInitialized property of the hotHalo component is settable.

tfamily \void isInitializedSet(<logical> value) Set the isInitialized property of the hotHalo component.

tfamily \textcolor{red}{\textless\textgreater} mass() Get the mass property of the hotHalo component.

tfamily \textcolor{red}{\textless\textgreater} massAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list off attributes for the mass property of the hotHalo component.

tfamily \textcolor{red}{\textless\textgreater} massCold() Get the massCold property of the hotHalo component.

tfamily \textcolor{red}{\textless\textgreater} massColdAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list off attributes for the massCold property of the hotHalo component.
tfamily \intzero massColdCount() Compute the count of evolvable quantities in the **massCold** property of the **HotHaloColdMode** component.

tfamily \textcolor{red}{\textless double\textgreater} massColdIsGettable() Get the **massCold** property of the **hotHalo** component.

tfamily \logicalzero massColdIsSettable() Specify whether the **massCold** property of the **hotHalo** component is settable.

tfamily \void massColdRate(<double> value) Cumulate to the rate of the **massCold** property of the **HotHaloColdMode** component.

tfamily \void massColdScale(<double> value) Set the scale of the **massCold** property of the **HotHaloColdMode** component.

tfamily \void massColdSet(<double> value) Set the **massCold** property of the **hotHalo** component.

tfamily \intzero massCount() Compute the count of evolvable quantities in the **mass** property of the **HotHaloStandard** component.

tfamily \textcolor{red}{\textless double\textgreater} massIsGettable() Get the **mass** property of the **hotHalo** component.

tfamily \logicalzero massIsSettable() Specify whether the **mass** property of the **hotHalo** component is settable.

tfamily \void massRate(<double> value) Cumulate to the rate of the **mass** property of the **HotHaloStandard** component.

tfamily \void massRemovalRate() Called whenever the standard hot halo component removes mass from the halo.

tfamily \void massScale(<double> value) Set the scale of the **mass** property of the **HotHaloStandard** component.

tfamily \void massSet(<double> value) Set the **mass** property of the **hotHalo** component.

tfamily \textcolor{red}{\textless double\textgreater} massSink() Get the **massSink** property of the **hotHalo** component.

tfamily \textcolor{red}{\textless type(varying\_string)(:\textgreater)} massSinkAttributeMatch(<logical> requireGettable \rightarrow <logical> requireSettable \rightarrow <logical> requireEvolvable \rightarrow) Return a list of implementations that provide the given list off attributes for the **massSink** property of the **hotHalo** component.

tfamily \intzero massSinkCount() Compute the count of evolvable quantities in the **massSink** property of the **HotHaloStandard** component.

tfamily \textcolor{red}{\textless double\textgreater} massSinkIsGettable() Get the **massSink** property of the **hotHalo** component.

tfamily \logicalzero massSinkIsSettable() Specify whether the **massSink** property of the **hotHalo** component is settable.

tfamily \void massSinkRate(<double> value) Cumulate to the rate of the **massSink** property of the **hotHalo** component.
16. Coding GALACTICUS

tfamily \void massSinkRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the massSink property of the HotHaloStandard component.

tfamily \logicalzero massSinkRateIsAttached() Return whether the rate method of the massSink property of the HotHaloStandard component has been attached to a function.

tfamily \void nameFromIndex(<integer> count →, <varying_string>name←) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the hotHalo component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \textcolor{red}{\textless double\textgreater} outerRadius() Get the outerRadius property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater\textgreater} outerRadiusAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the outerRadius property of the hotHalo component.

tfamily \intzero outerRadiusCount() Compute the count of evolvable quantities in the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusFunction(<function()> deferredFunction) Set the function to be used for the get method of the outerRadius property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater\textgreater} outerRadiusGrowthRate() Get the outerRadiusGrowthRate property of the HotHaloOutflowTracking component.

tfamily \void outerRadiusGrowthRateFunction(procedure(HotHaloOutflowTrackingouterRadiusGrowthRateInterface) deferredFunction) Set the function for the deferred outerRadiusGrowthRate property of the HotHaloOutflowTracking component.

tfamily \logicalzero outerRadiusGrowthRateFunctionIsSet() Specify whether the deferred function for the outerRadiusGrowthRate property of the HotHaloOutflowTracking component has been set.

tfamily \logicalzero outerRadiusIsAttached() Return whether the get method of the outerRadius property of the HotHaloStandard component has been attached to a function.

tfamily \textcolor{red}{\textless double\textgreater\textgreater} outerRadiusIsGettable() Get the outerRadius property of the hotHalo component.

tfamily \logicalzero outerRadiusIsSettable() Specify whether the outerRadius property of the hotHalo component is settable.

tfamily \void outerRadiusRate(<double> value) Cumulate to the rate of the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusScale(<double> value) Set the scale of the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusSet(<double> value) Set the outerRadius property of the hotHalo component.
16.5. Objects

tfamily \textcolor{red}{\textless \textgreater} \texttt{outerRadiusValue}() Get the \texttt{outerRadius} property of the \texttt{hotHalo} component.

tfamily \texttt{outflowedAbundances()} Get the \texttt{outflowedAbundances} property of the \texttt{hotHalo} component.

tfamily \texttt{outflowedAbundancesAttributeMatch\langle \texttt{logical}[\texttt{requireGettable}] \rightarrow, \texttt{logical}[\texttt{requireSettable}] \rightarrow, \texttt{logical}[\texttt{requireEvolvable}] \rightarrow\rangle} Return a list of implementations that provide the given list of attributes for the \texttt{outflowedAbundances} property of the \texttt{hotHalo} component.

tfamily \intzero \texttt{outflowedAbundancesCount()} Compute the count of evolvable quantities in the \texttt{outflowedAbundances} property of the \texttt{HotHaloStandard} component.

tfamily \textcolor{red}{\textless \textgreater} \texttt{outflowedAbundancesIsGettable}() Get the \texttt{outflowedAbundances} property of the \texttt{hotHalo} component.

tfamily \texttt{outflowedAbundancesIsSettable()} Specify whether the \texttt{outflowedAbundances} property of the \texttt{hotHalo} component is settable.

tfamily \texttt{outflowedAbundancesRate\langle \texttt{type(abundances)} > value\rangle} Cumulate to the rate of the \texttt{outflowedAbundances} property of the \texttt{HotHaloStandard} component.

tfamily \texttt{outflowedAbundancesScale\langle \texttt{type(abundances)} > value\rangle} Set the scale of the \texttt{outflowedAbundances} property of the \texttt{HotHaloStandard} component.

tfamily \texttt{outflowedAbundancesSet\langle \texttt{type(abundances)} > value\rangle} Set the \texttt{outflowedAbundances} property of the \texttt{hotHalo} component.

tfamily \textcolor{red}{\textless \textgreater} \texttt{outflowedAngularMomentum() Get the \texttt{outflowedAngularMomentum} property of the \texttt{hotHalo} component.

tfamily \texttt{outflowedAngularMomentumAttributeMatch\langle \texttt{logical}[\texttt{requireGettable}] \rightarrow, \texttt{logical}[\texttt{requireSettable}] \rightarrow, \texttt{logical}[\texttt{requireEvolvable}] \rightarrow\rangle} Return a list of implementations that provide the given list of attributes for the \texttt{outflowedAngularMomentum} property of the \texttt{hotHalo} component.

tfamily \intzero \texttt{outflowedAngularMomentumCount()} Compute the count of evolvable quantities in the \texttt{outflowedAngularMomentum} property of the \texttt{HotHaloStandard} component.

tfamily \textcolor{red}{\textless \textgreater} \texttt{outflowedAngularMomentumIsGettable}() Get the \texttt{outflowedAngularMomentum} property of the \texttt{hotHalo} component.

tfamily \texttt{outflowedAngularMomentumIsSettable()} Specify whether the \texttt{outflowedAngularMomentum} property of the \texttt{hotHalo} component is settable.

tfamily \texttt{outflowedAngularMomentumRate\langle \texttt{double} > value\rangle} Cumulate to the rate of the \texttt{outflowedAngularMomentum} property of the \texttt{HotHaloStandard} component.

tfamily \texttt{outflowedAngularMomentumScale\langle \texttt{double} > value\rangle} Set the scale of the \texttt{outflowedAngularMomentum} property of the \texttt{HotHaloStandard} component.

tfamily \texttt{outflowedAngularMomentumSet\langle \texttt{double} > value\rangle} Set the \texttt{outflowedAngularMomentum} property of the \texttt{hotHalo} component.
16. Coding GALACTICUS

tfamily {\textcolor{red}{\textless double\textgreater}} outflowedMass() Get the outflowedMass property of the hotHalo component.

tfamily {\textcolor{red}{\textless double\textgreater}} outflowedMassAttributeMatch(<logical>[requireGettable], <logical>[requireSettable], <logical>[requireEvolvable]) Return a list of implementations that provide the given list of attributes for the outflowedMass property of the hotHalo component.

tfamily {\textcolor{red}{\textless double\textgreater}} outflowedMassCount() Compute the count of evolvable quantities in the outflowedMass property of the HotHaloStandard component.

tfamily {\textcolor{red}{\textless double\textgreater}} outflowedMassIsGettable() Get the outflowedMass property of the hotHalo component.

tfamily {\textcolor{red}{\textless double\textgreater}} outflowedMassIsSettable() Specify whether the outflowedMass property of the hotHalo component is settable.

tfamily void outflowedMassRate(<double> value) Cumulate to the rate of the outflowedMass property of the HotHaloStandard component.

tfamily void outflowedMassScale(<double> value) Set the scale of the outflowedMass property of the HotHaloStandard component.

tfamily void outflowedMassSet(<double> value) Set the outflowedMass property of the hotHalo component.

tfamily {\textcolor{red}{\textless double\textgreater}} outflowingAngularMomentum() Get the outflowingAngularMomentum property of the hotHalo component.

tfamily {\textcolor{red}{\textless double\textgreater}} outflowingAngularMomentumRate(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingAngularMomentum property of the HotHaloStandard component.

tfamily {\textcolor{red}{\textless double\textgreater}} outflowingAngularMomentumRateIsAttached() Return whether the rate method of the outflowingAngularMomentum property of the HotHaloStandard component has been attached to a function.

tfamily {\textcolor{red}{\textless double\textgreater}} outflowingAngularMomentumIsGettable() Get the outflowingAngularMomentum property of the hotHalo component.
16.5. Objects

tfamily \textcolor{red}{\textless\textless\textgreater}\textless\textless\textgreater\textless\textless\textgreater} outflowingAngularMomentumAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→)
Return a list of implementations that provide the given list of attributes for the outflowingAngularMomentum property of the hotHalo component.

tfamily \intzero outflowingAngularMomentumCount() Compute the count of evolvable quantities in the outflowingAngularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textless\textgreater} outflowingAngularMomentumIsGettable() Get the outflowingAngularMomentum property of the hotHalo component.

tfamily \logicalzero outflowingAngularMomentumIsSettable() Specify whether the outflowingAngularMomentum property of the hotHalo component is settable.

tfamily \void outflowingAngularMomentumRate(<double> value) Cumulate to the rate of the outflowingAngularMomentum property of the hotHalo component.

tfamily \void outflowingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingAngularMomentum property of the HotHaloStandard component.

tfamily \logicalzero outflowingAngularMomentumRateIsAttached() Return whether the rate method of the outflowingAngularMomentum property of the HotHaloStandard component has been attached to a function.

tfamily \textcolor{red}{\textless\textless\textgreater} outflowingMassAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→)
Return a list of implementations that provide the given list of attributes for the outflowingMass property of the hotHalo component.

tfamily \intzero outflowingMassCount() Compute the count of evolvable quantities in the outflowingMass property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textless\textgreater} outflowingMassIsGettable() Get the outflowingMass property of the hotHalo component.

tfamily \logicalzero outflowingMassIsSettable() Specify whether the outflowingMass property of the hotHalo component is settable.

tfamily \void outflowingMassRate(<double> value) Cumulate to the rate of the outflowingMass property of the hotHalo component.

tfamily \void outflowingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingMass property of the HotHaloStandard component.

tfamily \logicalzero outflowingMassRateIsAttached() Return whether the rate method of the outflowingMass property of the HotHaloStandard component has been attached to a function.

tfamily \textcolor{red}{\textless\textless\textgreater} outflowReturn() Get the outflowReturn property of the HotHaloOutflowTracking component.

tfamily \void outflowReturnFunction(procedure(HotHaloOutflowTracking outflowReturnInterface) deferredFunction) Set the function for the deferred outflowReturn property of the HotHaloOutflowTracking component.
16. Coding GALACTICUS

tfamily \logicalzero outflowReturnFunctionIsSet() Specify whether the deferred function for the outflowReturn
property of the HotHaloOutflowTracking component has been set.

tfamily \logicalzero outflowTrackingIsActive() Return whether the outflowTracking implementation of
the hotHalo component class is active.

tfamily \void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(;;)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(;;)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \void outputCount(<integer>integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty↔, <char[*](::)> integerPropertyNames↔, <char[*](::)> integerPropertyComments↔, <double(:)> integerPropertyUnitsSI↔, <integer> doubleProperty↔, <char[*](::)> doublePropertyNames↔, <char[*](::)> doublePropertyComments↔, <double(;;)> doublePropertyUnitsSI↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \doublezero potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

tfamily \void readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the
given fileHandle.

tfamily \doublezero rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(;;)> array→) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(;;)> array→) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(;;)> array→) Serialize the evolvable quantities to an array.

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the hotHalo
component class is active.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} strippedAbundances() Get the strippedAbundances
property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(::)\textgreater} strippedAbundancesAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the strippedAbundances property of the hotHalo component.

tfamily \intzero strippedAbundancesCount() Compute the count of evolvable quantities in the strippedAbundances
property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} strippedAbundancesIsGettable() Get the strippedAbundances property of the hotHalo component.
tfamily \logicalzero strippedAbundancesIsSettable() Specify whether the **strippedAbundances** property of the **hotHalo** component is settable.

tfamily \void strippedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the **strippedAbundances** property of the **HotHaloStandard** component.

tfamily \void strippedAbundancesScale(<type(abundances)> value) Set the scale of the **strippedAbundances** property of the **HotHaloStandard** component.

tfamily \void strippedAbundancesSet(<type(abundances)> value) Set the **strippedAbundances** property of the **hotHalo** component.

tfamily \textcolor{red}{\textless double\textgreater} strippedMass() Get the **strippedMass** property of the **hotHalo** component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)} strippedMassAttributeMatch(<logical> requireGettable → , <logical> requireSettable → , <logical> requireEvolvable → ) Return a list of implementations that provide the given list off attributes for the **strippedMass** property of the **hotHalo** component.

tfamily \intzero strippedMassCount() Compute the count of evolvable quantities in the **strippedMass** property of the **HotHaloStandard** component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} trackedOutflowAbundances() Get the **trackedOutflowAbundances** property of the **hotHalo** component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)} trackedOutflowAbundancesAttributeMatch(<logical> requireGettable → , <logical> requireSettable → , <logical> requireEvolvable → ) Return a list of implementations that provide the given list off attributes for the **trackedOutflowAbundances** property of the **hotHalo** component.

tfamily \intzero trackedOutflowAbundancesCount() Compute the count of evolvable quantities in the **trackedOutflowAbundances** property of the **HotHaloOutflowTracking** component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} trackedOutflowAbundancesIsGettable() Get the **trackedOutflowAbundances** property of the **hotHalo** component.
tfamily \logicalzero trackedOutflowAbundancesIsSettable() Specify whether the trackedOutflowAbundances property of the hotHalo component is settable.

tfamily \void trackedOutflowAbundancesRate(<type(abundances)> value) Cumulate to the rate of the trackedOutflowAbundances property of the HotHaloOutflowTracking component.

tfamily \void trackedOutflowAbundancesScale(<type(abundances)> value) Set the scale of the trackedOutflowAbundances property of the HotHaloOutflowTracking component.

tfamily \void trackedOutflowAbundancesSet(<type(abundances)> value) Set the trackedOutflowAbundances property of the hotHalo component.

```
tfamily \textcolor{red}{\textless double\textgreater} trackedOutflowMass() Get the trackedOutflowMass property of the hotHalo component.
```

```
tfamily \textcolor{red}{\textless type(varying\_string):)\textgreater} trackedOutflowMassAttributeMatch(<logical>[requireGettable]\rightarrow, <logical>[requireSettable]\rightarrow, <logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the trackedOutflowMass property of the hotHalo component.
```

```
tfamily \intzero trackedOutflowMassCount() Compute the count of evolvable quantities in the trackedOutflowMass property of the HotHaloOutflowTracking component.
```

```
tfamily \textcolor{red}{\textless double\textgreater} unaccretedMass() Get the unaccretedMass property of the hotHalo component.
```

```
tfamily \textcolor{red}{\textless type(varying\_string):)\textgreater} unaccretedMassAttributeMatch(<logical>[requireGettable]\rightarrow, <logical>[requireSettable]\rightarrow, <logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the unaccretedMass property of the hotHalo component.
```

```
tfamily \intzero unaccretedMassCount() Compute the count of evolvable quantities in the unaccretedMass property of the HotHaloStandard component.
```

```
tfamily \textcolor{red}{\textless double\textgreater} unaccretedMassIsGettable() Get the unaccretedMass property of the hotHalo component.
```

```
tfamily \logicalzero unaccretedMassIsSettable() Specify whether the unaccretedMass property of the hotHalo component is settable.
```

```
tfamily \textcolor{red}{\textless type(varying\_string):)\textgreater} type() Return the type of this object.
```

```
tfamily \textcolor{red}{\textless double\textgreater} unaccretedMass() Get the unaccretedMass property of the hotHalo component.
```

```
tfamily \textcolor{red}{\textless type(varying\_string):)\textgreater} unaccretedMassAttributeMatch(<logical>[requireGettable]\rightarrow, <logical>[requireSettable]\rightarrow, <logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the unaccretedMass property of the hotHalo component.
```

```
tfamily \intzero unaccretedMassCount() Compute the count of evolvable quantities in the unaccretedMass property of the HotHaloStandard component.
```

```
tfamily \textcolor{red}{\textless double\textgreater} unaccretedMassIsGettable() Get the unaccretedMass property of the hotHalo component.
```

```
tfamily \logicalzero unaccretedMassIsSettable() Specify whether the unaccretedMass property of the hotHalo component is settable.
```

540
16.5. Objects

tfamily \void unaccretedMassRate(<double> value) Cumulate to the rate of the unaccretedMass property of the HotHaloStandard component.

tfamily \void unaccretedMassScale(<double> value) Set the scale of the unaccretedMass property of the HotHaloStandard component.

tfamily \void unaccretedMassSet(<double> value) Set the unaccretedMass property of the hotHalo component.

tfamily \logicalzero verySimpleIsActive() Return whether the verySimple implementation of the hotHalo component class is active.

tfamily nodeComponentHotHaloStandard

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundances() Get the abundances property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater abundancesAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the abundances property of the hotHalo component

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesCold() Get the abundancesCold property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater abundancesColdAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the abundancesCold property of the hotHalo component

tfamily \intzero abundancesColdCount() Compute the count of evolvable quantities in the abundancesCold property of the HotHaloColdMode component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesColdIsGettable() Get the abundancesCold property of the hotHalo component.

tfamily \logicalzero abundancesColdIsSettable() Specify whether the abundancesCold property of the hotHalo component is settable.

tfamily \void abundancesColdRate(<type(abundances)> value) Cumulate to the rate of the abundancesCold property of the HotHaloColdMode component.

tfamily \void abundancesColdScale(<type(abundances)> value) Set the scale of the abundancesCold property of the HotHaloColdMode component.

tfamily \void abundancesColdSet(<type(abundances)> value) Set the abundancesCold property of the hotHalo component.

tfamily \intzero abundancesCount() Compute the count of evolvable quantities in the abundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesIsGettable() Get the abundances property of the hotHalo component.

tfamily \logicalzero abundancesIsSettable() Specify whether the abundances property of the hotHalo component is settable.
tfamily \textcolor{red}{\textless \text{double}\textgreater} angularMomentum() Get the angularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless \text{type(abundances)}\textgreater} abundancesRate(<\text{type(abundances)}> value) Cumulate to the rate of the abundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless \text{type(abundances)}\textgreater} abundancesScale(<\text{type(abundances)}> value) Set the scale of the abundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless \text{type(abundances)}\textgreater} abundancesSet(<\text{type(abundances)}> value) Set the abundances property of the hotHalo component.

tfamily \textcolor{red}{\textless \text{double}\textgreater} angularMomentumCold() Get the angularMomentumCold property of the hotHalo component.

tfamily \textcolor{red}{\textless \text{type(varying\_string)}\textgreater} angularMomentumColdAttributeMatch(<\text{logical}> requireGettable\rightarrow, <\text{logical}> requireSettable\rightarrow, <\text{logical}> requireEvolvable\rightarrow) Return a list of implementations that provide the given list of attributes for the angularMomentumCold property of the hotHalo component.

tfamily \textcolor{red}{\textless \text{double}\textgreater} angularMomentumColdIsGettable() Get the angularMomentumCold property of the hotHalo component.

tfamily \textcolor{red}{\textless \text{logical}\textgreater} angularMomentumColdIsSettable() Specify whether the angularMomentumCold property of the hotHalo component is settable.

tfamily \int zero angularMomentumColdCount() Compute the count of evolvable quantities in the angularMomentumCold property of the HotHaloColdMode component.

tfamily \textcolor{red}{\textless \text{double}\textgreater} angularMomentumIsGettable() Get the angularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless \text{logical}\textgreater} angularMomentumIsSettable() Specify whether the angularMomentum property of the hotHalo component is settable.

tfamily \void angularMomentumRate(<\text{double}> value) Cumulate to the rate of the angularMomentum property of the HotHaloStandard component.

tfamily \void angularMomentumScale(<\text{double}> value) Set the scale of the angularMomentum property of the HotHaloColdMode component.

tfamily \void angularMomentumSet(<\text{double}> value) Set the angularMomentum property of the hotHalo component.

tfamily \int zero angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless \text{double}\textgreater} angularMomentumIsGettable() Get the angularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless \text{logical}\textgreater} angularMomentumIsSettable() Specify whether the angularMomentum property of the hotHalo component is settable.
16.5. Objects

tfamily \void angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the HotHaloStandard component.

tfamily \void angularMomentumSet(<double> value) Set the angularMomentum property of the hotHalo component.

tfamily \void builder(<*type(node)> componentDefinition) Build a nodeComponent from a supplied XML definition.

\textcolor{red}{\lesstype(chemicalAbundances) \greater} chemicals() Get the chemicals property of the hotHalo component.

\textcolor{red}{\lesstype(varying\_string) (: \greater)} chemicalsAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the chemicals property of the hotHalo component.

intzero chemicalsCount() Compute the count of evolvable quantities in the chemicals property of the HotHaloStandard component.

\textcolor{red}{\lesstype(chemicalAbundances) \greater} chemicalsIsGettable() Get the chemicals property of the hotHalo component.

logicalzero chemicalsIsSettable() Specify whether the chemicals property of the hotHalo component is settable.

\void chemicalsRate(<type(chemicalAbundances)> value) Cumulate to the rate of the chemicals property of the HotHaloStandard component.

\void chemicalsScale(<type(chemicalAbundances)> value) Set the scale of the chemicals property of the HotHaloStandard component.

\void chemicalsSet(<type(chemicalAbundances)> value) Set the chemicals property of the hotHalo component.

logicalzero coldModeIsActive() Return whether the coldMode implementation of the hotHalo component class is active.

\void createFunctionSet(<function()>) Set the function used to create HotHaloStandard components.

density(<double(3)> positionSpherical →, <componentType>[componentType] →, <massType>[massType] →, <weightBy>[weightBy] →, <integer>[weightIndex] →, <logical>[haloLoaded] →) Compute the density.

deserializeRates(<double(:)> array ←) Deserialize the evolvable rates from an array.

deserializeScales(<double(:)> array ←) Deserialize the evolvable scales from an array.

deserializeValues(<double(:)> array ←) Deserialize the evolvable quantities from an array.

destroy() Destroy the object.

dump() Generate an ASCII dump of all properties.

dumpRaw(<integer> fileHandle →) Generate a binary dump of all properties.

dumpXML() Generate an XML dump of all properties.
16. Coding GALACTICUS

tfamily double zero enclosedMass("double") radius\rightarrow, "componentType" [componentType]\rightarrow, "massType" [massType]\rightarrow, "weightBy" [weightBy]\rightarrow, "integer" [weightIndex]\rightarrow, "logical" [haloLoaded]\rightarrow) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless double\textgreater} heatSource() Get the \texttt{heatSource} property of the \texttt{hotHalo} component.

tfamily \textcolor{red}{\textless textless double\textgreater\textgreater} heatSourceAttributeMatch("logical" [requireGettable]\rightarrow, "logical" [requireSettable]\rightarrow, "logical" [requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the \texttt{heatSource} property of the \texttt{hotHalo} component

tfamily \intzero heatSourceCount() Compute the count of evolvable quantities in the \texttt{heatSource} property of the \texttt{HotHaloStandard} component.

tfamily \textcolor{red}{\textless textless double\textgreater\textgreater} heatSourceIsGettable() Get the \texttt{heatSource} property of the \texttt{hotHalo} component.

ntfamily \logicalzero heatSourceIsSettable() Specify whether the \texttt{heatSource} property of the \texttt{hotHalo} component is settable.

tfamily \void heatSourceRate("double" value) Cumulate to the rate of the \texttt{heatSource} property of the \texttt{hotHalo} component.

tfamily \void heatSourceRateFunction("function()" deferredFunction) Set the function to be used for the \texttt{rate} method of the \texttt{heatSource} property of the \texttt{HotHaloStandard} component.

ntfamily \logicalzero heatSourceRateIsAttached() Return whether the rate method of the \texttt{heatSource} property of the \texttt{HotHaloStandard} component has been attached to a function.

ntfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host \texttt{treeNode} object.

ntfamily \textcolor{red}{\textless type(abundances)\textgreater} hotHaloCoolingAbundances() Get the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component.

ntfamily \textcolor{red}{\textless textless type(varying\_string)()\textgreater\textgreater} hotHaloCoolingAbundancesAttributeMatch("logical" [requireGettable]\rightarrow, "logical" [requireSettable]\rightarrow, "logical" [requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component

ntfamily \intzero hotHaloCoolingAbundancesCount() Compute the count of evolvable quantities in the \texttt{hotHaloCoolingAbundances} property of the \texttt{HotHaloStandard} component.

ntfamily \textcolor{red}{\textless type(abundances)\textgreater\textgreater} hotHaloCoolingAbundancesIsGettable() Get the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component.

ntfamily \logicalzero hotHaloCoolingAbundancesIsSettable() Specify whether the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component is settable.

ntfamily \void hotHaloCoolingAbundancesRate("type(abundances)" value) Cumulate to the rate of the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component.

ntfamily \void hotHaloCoolingAbundancesRateFunction("function()" deferredFunction) Set the function to be used for the \texttt{rate} method of the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component.
16.5. Objects

tfamily \logicalzero hotHaloCoolingAbundancesRateIsAttached() Return whether the rate method of the hotHaloCoolingAbundances property of the hotHalo component has been attached to a function.

tfamily \textcolor{red}\textless double\textgreater hotHaloCoolingAngularMomentum() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}\textless double\textgreater hotHaloCoolingAngularMomentumIsGettable() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \logicalzero hotHaloCoolingAngularMomentumIsSettable() Specify whether the hotHaloCoolingAngularMomentum property of the hotHalo component is settable.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \logicalzero hotHaloCoolingAngularMomentumRateIsAttached() Return whether the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component has been attached to a function.

tfamily \textcolor{red}\textless double\textgreater hotHaloCoolingMass() Get the hotHaloCoolingMass property of the hotHalo component.

tfamily \textcolor{red}\textless double\textgreater hotHaloCoolingMassIsGettable() Get the hotHaloCoolingMass property of the hotHalo component.

tfamily \logicalzero hotHaloCoolingMassIsSettable() Specify whether the hotHaloCoolingMass property of the hotHalo component is settable.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void hotHaloCoolingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingMass property of the hotHalo component.
16. Coding GALACTICUS

tfamily \logicalzero hotHaloCoolingMassRateIsAttached() Return whether the rate method of the hotHalo-CoolingMass property of the hotHalo component has been attached to a function.

tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}{\textless \textless \textgreater \textless \textgreater} isInitialized() Get the isInitialized property of the hotHalo component.

tfamily \textcolor{red}{\textless \textless \textgreater \textless \textgreater} isInitializedAttributeMatch(\logical[requireGettable]→, \logical[requireSettable]→, \logical[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the isInitialized property of the hotHalo component.

tfamily \logicalzero isInitializedIsGettable() Get the isInitialized property of the hotHalo component.

tfamily \logicalzero isInitializedIsSettable() Specify whether the isInitialized property of the hotHalo component is settable.

tfamily \void isInitializedSet(\logical value) Set the isInitialized property of the hotHalo component.

tfamily \textcolor{red}{\textless \textless \textgreater \textless \textgreater} mass() Get the mass property of the hotHalo component.

tfamily \textcolor{red}{\textless \textless \textgreater \textless \textgreater} massAttributeMatch(\logical[requireGettable]→, \logical[requireSettable]→, \logical[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the mass property of the hotHalo component.

tfamily \textcolor{red}{\textless \textless \textgreater \textless \textgreater} massCold() Get the massCold property of the hotHalo component.

tfamily \textcolor{red}{\textless \textless \textgreater \textless \textgreater} massColdAttributeMatch(\logical[requireGettable]→, \logical[requireSettable]→, \logical[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the massCold property of the hotHalo component.

tfamily \intzero massColdCount() Compute the count of evolvable quantities in the massCold property of the HotHaloColdMode component.

tfamily \logicalzero massColdIsGettable() Get the massCold property of the hotHalo component.

tfamily \logicalzero massColdIsSettable() Specify whether the massCold property of the hotHalo component is settable.

tfamily \void massColdRate(\double value) Cumulate to the rate of the massCold property of the HotHaloColdMode component.

tfamily \void massColdScale(\double value) Set the scale of the massCold property of the HotHaloColdMode component.

tfamily \void massColdSet(\double value) Set the massCold property of the hotHalo component.

tfamily \intzero massCount() Compute the count of evolvable quantities in the mass property of the HotHaloStandard component.
16.5. Objects

tfamily \textcolor{red}{\textless double\textgreater} massIsGettable() Get the mass property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} massSink() Get the massSink property of the hotHalo component.

tfamily \logicalzero massIsSettable() Specify whether the mass property of the hotHalo component is settable.

tfamily \void massRate(<double> value) Cumulate to the rate of the mass property of the HotHaloStandard component.

tfamily \void massRemovalRate() Called whenever the standard hot halo component removes mass from the halo.

tfamily \void massScale(<double> value) Set the scale of the mass property of the HotHaloStandard component.

tfamily \void massSet(<double> value) Set the mass property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} massSink() Get the massSink property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype(varying_string)(:)} massSinkAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the massSink property of the hotHalo component.

tfamily \intzero massSinkCount() Compute the count of evolvable quantities in the massSink property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} massSinkIsGettable() Get the massSink property of the hotHalo component.

tfamily \logicalzero massSinkIsSettable() Specify whether the massSink property of the hotHalo component is settable.

tfamily \void massSinkRate(<double> value) Cumulate to the rate of the massSink property of the hotHalo component.

tfamily \void massSinkRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the massSink property of the HotHaloStandard component.

tfamily \logicalzero massSinkRateIsAttached() Return whether the rate method of the massSink property of the HotHaloStandard component has been attached to a function.

tfamily \void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the hotHalo component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \textcolor{red}{\textless double\textgreater} outerRadius() Get the outerRadius property of the hotHalo component.
16. Coding Galacticus

tfamily \textcolor{red}{\text<textless type(varying\string):\textgreater>} outerRadiusAttributeMatch(<logical>[requireGettable] →<logical>[requireSettable] →<logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the outerRadius property of the hotHalo component.

tfamily \intzero outerRadiusCount() Compute the count of evolvable quantities in the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusFunction(<function()> deferredFunction) Set the function to be used for the get method of the outerRadius property of the HotHaloStandard component.

tfamily \textcolor{red}{\text<textless double\textgreater>} outerRadiusGrowthRate() Get the outerRadiusGrowthRate property of the HotHaloStandard component.

tfamily \void outerRadiusGrowthRateFunction(procedure(HotHaloStandard outerRadiusGrowthRateInterface) deferredFunction) Set the function for the deferred outerRadiusGrowthRate property of the HotHaloStandard component.

tfamily \logical zero outerRadiusGrowthRateFunctionIsSet() Specify whether the deferred function for the outerRadiusGrowthRate property of the HotHaloStandard component has been set.

tfamily \logical zero outerRadiusIsAttached() Return whether the get method of the outerRadius property of the HotHaloStandard component has been attached to a function.

tfamily \textcolor{red}{\text<textless double\textgreater>} outerRadiusIsGettable() Get the outerRadius property of the hotHalo component.

tfamily \logical zero outerRadiusIsSettable() Specify whether the outerRadius property of the hotHalo component is settable.

tfamily \void outerRadiusRate(<double> value) Cumulate to the rate of the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusScale(<double> value) Set the scale of the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusSet(<double> value) Set the outerRadius property of the hotHalo component.

tfamily \textcolor{red}{\text<textless type(abundances)\textgreater>} outflowedAbundances() Get the outflowedAbundances property of the hotHalo component.

tfamily \textcolor{red}{\text<textless type(varying\string):\textgreater>} outflowedAbundancesAttributeMatch(<logical>[requireGettable] →<logical>[requireSettable] →<logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the outflowedAbundances property of the hotHalo component.

tfamily \intzero outflowedAbundancesCount() Compute the count of evolvable quantities in the outflowedAbundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\text<textless type(abundances)\textgreater>} outflowedAbundancesIsGettable() Get the outflowedAbundances property of the hotHalo component.
tfamily \textcolor{red}{\textless{}double\textgreater{}} outflowedAngularMomentum() Get the outflowedAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless{}double\textgreater{}} outflowedAngularMomentumIsGettable() Get the outflowedAngularMomentum property of the hotHalo component.

tfamily \logicalzero outflowedAngularMomentumIsSettable() Specify whether the outflowedAngularMomentum property of the hotHalo component is settable.

tfamily \void outflowedAngularMomentumRate(<double> value) Cumulate to the rate of the outflowedAngularMomentum property of the HotHaloStandard component.

tfamily \void outflowedAngularMomentumScale(<double> value) Set the scale of the outflowedAngularMomentum property of the HotHaloStandard component.

tfamily \void outflowedAngularMomentumSet(<double> value) Set the outflowedAngularMomentum property of the hotHalo component.

tfamily \intzero outflowedAngularMomentumCount() Compute the count of evolvable quantities in the outflowedAngularMomentum property of the HotHaloStandard component.

16.5. Objects

tfamily \logicalzero outflowedAbundancesIsSettable() Specify whether the outflowedAbundances property of the hotHalo component is settable.

tfamily \void outflowedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the outflowedAbundances property of the HotHaloStandard component.

tfamily \void outflowedAbundancesScale(<type(abundances)> value) Set the scale of the outflowedAbundances property of the HotHaloStandard component.

tfamily \void outflowedAbundancesSet(<type(abundances)> value) Set the outflowedAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless{}double\textgreater{}} outflowedAngularMomentum() Get the outflowedAngularMomentum property of the hotHalo component.

549
16. Coding GALACTICUS

tfamily \void outflowedMassScale(<double> value) Set the scale of the outflowedMass property of the HotHaloStandard component.

tfamily \void outflowedMassSet(<double> value) Set the outflowedMass property of the hotHalo component.

tfamily \textcolor{red}{\textless\textstring\textgreater} outflowingAbundances() Get the outflowingAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless\textstring\textgreater} outflowingAbundancesAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the outflowingAbundances property of the hotHalo component.

tfamily \intzero outflowingAbundancesCount() Compute the count of evolvable quantities in the outflowingAbundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textstring\textgreater} outflowingAbundancesIsGettable() Get the outflowingAbundances property of the hotHalo component.

tfamily \logicalzero outflowingAbundancesIsSettable() Specify whether the outflowingAbundances property of the hotHalo component is settable.

tfamily \void outflowingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the outflowingAbundances property of the hotHalo component.

tfamily \void outflowingAbundancesRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingAbundances property of the HotHaloStandard component.

tfamily \logicalzero outflowingAbundancesRateIsAttached() Return whether the rate method of the outflowingAbundances property of the HotHaloStandard component has been attached to a function.

tfamily \textcolor{red}{\textless\textstring\textgreater} outflowingAngularMomentum() Get the outflowingAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless\textstring\textgreater} outflowingAngularMomentumAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the outflowingAngularMomentum property of the hotHalo component.

tfamily \intzero outflowingAngularMomentumCount() Compute the count of evolvable quantities in the outflowingAngularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless\textstring\textgreater} outflowingAngularMomentumIsGettable() Get the outflowingAngularMomentum property of the hotHalo component.

tfamily \logicalzero outflowingAngularMomentumIsSettable() Specify whether the outflowingAngularMomentum property of the hotHalo component is settable.

tfamily \void outflowingAngularMomentumRate(<double> value) Cumulate to the rate of the outflowingAngularMomentum property of the hotHalo component.

tfamily \void outflowingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingAngularMomentum property of the HotHaloStandard component.
16.5. Objects

tfamily \logicalzero outflowingAngularMomentumRateIsAttached() Return whether the rate method of the outflowingAngularMomentum property of the HotHaloStandard component has been attached to a function.

tfamily \textcolor{red}{\textless double\textgreater} outflowingMass() Get the outflowingMass property of the hotHalo component.

tfamily \textcolor{red}{\textless void\textgreater} outflowReturn() Get the outflowReturn property of the HotHaloStandard component.

tfamily outflowingMassAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the outflowingMass property of the hotHalo component.

tfamily \intzerooutflowingMassCount() Compute the count of evolvable quantities in the outflowingMass property of the HotHaloStandard component.

tfamily \logicalzero outflowingMassIsGettable() Get the outflowingMass property of the hotHalo component.

tfamily \logicalzero outflowingMassIsSettable() Specify whether the outflowingMass property of the hotHalo component is settable.

tfamily \void outflowingMassRate(<double> value) Cumulate to the rate of the outflowingMass property of the hotHalo component.

tfamily \void outflowingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingMass property of the HotHaloStandard component.

tfamily \logicalzero outflowingMassRateIsAttached() Return whether the rate method of the outflowingMass property of the HotHaloStandard component has been attached to a function.

tfamily \void outputCount(<integer>integerPropertyCount ↔, <integer>doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily \void outputNames(<integer>integerProperty ↔, <char[*]>integerPropertyNames ↔, <char[*]>integerPropertyComments ↔, <double[*]>integerPropertyUnitsSI ↔, <integer>doubleProperty ↔, <char[*]>doublePropertyNames ↔, <char[*]>doublePropertyComments ↔, <double[*]>doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

tfamily \doublezero potential(<double> radius →, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the gravitational potential.
16. Coding Galacticus

tfamily \void readRaw(<integer> fileHandle \rightarrow) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \double zero rotationCurve(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the rotation curve.

tfamily \double zero rotationCurveGradient(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the rotation curve gradient.

tfamily \int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array \rightarrow) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array \rightarrow) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array \rightarrow) Serialize the evolvable quantities to an array.

tfamily \logical zero standardIsActive() Return whether the standard implementation of the hotHalo component class is active.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} strippedAbundances() Get the strippedAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} strippedAbundancesAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the strippedAbundances property of the hotHalo component.

tfamily \int zero strippedAbundancesCount() Compute the count of evolvable quantities in the strippedAbundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} strippedMass() Get the strippedMass property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} strippedMassAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the strippedMass property of the hotHalo component.

tfamily \int zero strippedMassCount() Compute the count of evolvable quantities in the strippedMass property of the HotHaloStandard component.
16.5. Objects

tfamily \textcolor{red}{\textless double\textgreater} strippedMassIsGettable() Get the strippedMass property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} strippedMassIsSettable() Specify whether the strippedMass property of the hotHalo component is settable.

tfamily \textcolor{red}{\textless double\textgreater} strippedMassRate(<double> value) Cumulate to the rate of the strippedMass property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} strippedMassScale(<double> value) Set the scale of the strippedMass property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} strippedMassSet(<double> value) Set the strippedMass property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} trackedOutflowAbundances() Get the trackedOutflowAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} trackedOutflowAbundancesAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the trackedOutflowAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} trackedOutflowAbundancesCount() Compute the count of evolvable quantities in the trackedOutflowAbundances property of the HotHaloOutflowTracking component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} trackedOutflowAbundancesIsGettable() Get the trackedOutflowAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} trackedOutflowAbundancesIsSettable() Specify whether the trackedOutflowAbundances property of the hotHalo component is settable.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} trackedOutflowAbundancesRate(<type(abundances)> value) Cumulate to the rate of the trackedOutflowAbundances property of the HotHaloOutflowTracking component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} trackedOutflowAbundancesScale(<type(abundances)> value) Set the scale of the trackedOutflowAbundances property of the HotHaloOutflowTracking component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} trackedOutflowAbundancesSet(<type(abundances)> value) Set the trackedOutflowAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} trackedOutflowMass() Get the trackedOutflowMass property of the hotHalo component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} trackedOutflowMassAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the trackedOutflowMass property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} trackedOutflowMassCount() Compute the count of evolvable quantities in the trackedOutflowMass property of the HotHaloOutflowTracking component.
16. Coding GALACTICUS

tfamily \textcolor{red}{\textless\textless double\textgreater\textgreater} trackedOutflowMassIsGettable() Get the trackedOutflowMass property of the hotHalo component.

tfamily \textcolor{red}{\logicalzero} trackedOutflowMassIsSettable() Specify whether the trackedOutflowMass property of the hotHalo component is settable.

tfamily \void trackedOutflowMassRate(\double value) Cumulate to the rate of the trackedOutflowMass property of the HotHaloOutflowTracking component.

tfamily \void trackedOutflowMassScale(\double value) Set the scale of the trackedOutflowMass property of the HotHaloOutflowTracking component.

tfamily \void trackedOutflowMassSet(\double value) Set the trackedOutflowMass property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

tfamily \textcolor{red}{\textless double\textgreater} unaccretedMass() Get the unaccretedMass property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)\textgreater} unaccretedMassAttributeMatch(\logical requirementGettable\rightarrow, \logical requirementSettable\rightarrow, \logical requirementEvolvable\rightarrow) Return a list of implementations that provide the given list of attributes for the unaccretedMass property of the hotHalo component.

tfamily \intzero unaccretedMassCount() Compute the count of evolvable quantities in the unaccretedMass property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} unaccretedMassIsGettable() Get the unaccretedMass property of the hotHalo component.

tfamily \textcolor{red}{\logicalzero} unaccretedMassIsSettable() Specify whether the unaccretedMass property of the hotHalo component is settable.

tfamily \void unaccretedMassRate(\double value) Cumulate to the rate of the unaccretedMass property of the HotHaloStandard component.

tfamily \void unaccretedMassScale(\double value) Set the scale of the unaccretedMass property of the HotHaloStandard component.

tfamily \void unaccretedMassSet(\double value) Set the unaccretedMass property of the hotHalo component.

tfamily \textcolor{red}{\logicalzero} verySimpleIsActive() Return whether the verySimple implementation of the hotHalo component class is active.

tfamily nodeComponentHotHaloVerySimple

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundances() Get the abundances property of the hotHalo component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)\textgreater} abundancesAttributeMatch(\logical requirementGettable\rightarrow, \logical requirementSettable\rightarrow, \logical requirementEvolvable\rightarrow) Return a list of implementations that provide the given list of attributes for the abundances property of the hotHalo component.
16.5. Objects

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesCold() Get the abundancesCold property of the hotHalo component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesColdAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the abundancesCold property of the hotHalo component.

tfamily \intzero abundancesColdCount() Compute the count of evolvable quantities in the abundancesCold property of the HotHaloColdMode component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesColdIsGettable() Get the abundancesCold property of the hotHalo component.

tfamily \logicalzero abundancesColdIsSettable() Specify whether the abundancesCold property of the hotHalo component is settable.

tfamily \void abundancesColdRate(<type(abundances)> value) Cumulate to the rate of the abundancesCold property of the HotHaloColdMode component.

tfamily \void abundancesColdScale(<type(abundances)> value) Set the scale of the abundancesCold property of the HotHaloColdMode component.

tfamily \void abundancesColdSet(<type(abundances)> value) Set the abundancesCold property of the hotHalo component.

tfamily \intzero abundancesCount() Compute the count of evolvable quantities in the abundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesIsGettable() Get the abundances property of the hotHalo component.

tfamily \logicalzero abundancesIsSettable() Specify whether the abundances property of the hotHalo component is settable.

tfamily \void abundancesRate(<type(abundances)> value) Cumulate to the rate of the abundances property of the HotHaloStandard component.

tfamily \void abundancesScale(<type(abundances)> value) Set the scale of the abundances property of the HotHaloStandard component.

tfamily \void abundancesSet(<type(abundances)> value) Set the abundances property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentum() Get the angularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} angularMomentumAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the angularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumCold() Get the angularMomentumCold property of the hotHalo component.
16. Coding GALACTICUS

tfamily \textcolor{red}{\textless{}textless type(varying\_\_string):(::)}\textgreater{} angularMomentumColdAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the angularMomentumCold property of the hotHalo component.

tfamily \intzero angularMomentumColdCount() Compute the count of evolvable quantities in the angularMomentumCold property of the HotHaloColdMode component.

tfamily \textcolor{red}{\textless{}double\textgreater{}} angularMomentumColdIsGettable() Get the angularMomentumCold property of the hotHalo component.

tfamily \logicalzero angularMomentumColdIsSettable() Specify whether the angularMomentumCold property of the hotHalo component is settable.

tfamily \void angularMomentumColdRate(<double> value) Cumulate to the rate of the angularMomentumCold property of the HotHaloColdMode component.

tfamily \void angularMomentumColdScale(<double> value) Set the scale of the angularMomentumCold property of the HotHaloColdMode component.

tfamily \void angularMomentumColdSet(<double> value) Set the angularMomentumCold property of the hotHalo component.

tfamily \intzero angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless{}textless double\textgreater{}} angularMomentumIsGettable() Get the angularMomentum property of the hotHalo component.

tfamily \logicalzero angularMomentumIsSettable() Specify whether the angularMomentum property of the hotHalo component is settable.

tfamily \void angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the HotHaloStandard component.

tfamily \void angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the HotHaloStandard component.

tfamily \void angularMomentumSet(<double> value) Set the angularMomentum property of the hotHalo component.

tfamily \void builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

 elemental \textcolor{red}{\textless{}textless type(chemicalAbundances):(::)}\textgreater{} chemicals() Get the chemicals property of the hotHalo component.

tfamily \textcolor{red}{\textless{}textless type(varying\_\_string):(::)}\textgreater{} chemicalsAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the chemicals property of the hotHalo component.

tfamily \intzero chemicalsCount() Compute the count of evolvable quantities in the chemicals property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless{}textless type(chemicalAbundances):(::)}\textgreater{} chemicalsIsGettable() Get the chemicals property of the hotHalo component.
16.5. Objects

tfamily logicalzero chemicalsIsSettable() Specify whether the chemicals property of the hotHalo component is settable.

tfamily void chemicalsRate(<type(chemicalAbundances)> value) Cumulate to the rate of the chemicals property of the HotHaloStandard component.

tfamily void chemicalsScale(<type(chemicalAbundances)> value) Set the scale of the chemicals property of the HotHaloStandard component.

tfamily void chemicalsSet(<type(chemicalAbundances)> value) Set the chemicals property of the hotHalo component.

tfamily logicalzero coldModeIsActive() Return whether the coldMode implementation of the hotHalo component class is active.

tfamily double zero density(<double(3)> positionSpherical →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the density.

tfamily void deserializeRates(<double(:)> array ←) Deserialize the evolvable rates from an array.

tfamily void deserializeScales(<double(:)> array ←) Deserialize the evolvable scales from an array.

tfamily void deserializeValues(<double(:)> array ←) Deserialize the evolvable quantities from an array.

tfamily void destroy() Destroy the object.

tfamily void dump() Generate an ASCII dump of all properties.

tfamily void dumpRaw(<integer> fileHandle →) Generate a binary dump of all properties.

tfamily void dumpXML() Generate an XML dump of all properties.

tfamily double zero enclosedMass(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the mass enclosed within a radius.

tfamily textcolor{red} \{\textless\textless\textless double\textgreater\textgreater\} heatSource() Get the heatSource property of the hotHalo component.

tfamily textcolor{red} \{\textless\textless\textless type(varying\_string)(:)\textgreater\textgreater\} heatSourceAttributeMatch(<logical> [requireGettable] →, <logical> [requireSettable] →, <logical> [requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the heatSource property of the hotHalo component.

tfamily int zero heatSourceCount() Compute the count of evolvable quantities in the heatSource property of the HotHaloStandard component.

tfamily textcolor{red} \{\textless\textless\textless double\textgreater\textgreater\} heatSourceIsGettable() Get the heatSource property of the hotHalo component.

tfamily logicalzero heatSourceIsSettable() Specify whether the heatSource property of the hotHalo component is settable.

tfamily void heatSourceRate(<double> value) Cumulate to the rate of the heatSource property of the HotHaloStandard component.
16. Coding GALACTICUS

tfamily \textcolor{red}\textless *treeNode\textgreater \ host() Return a pointer to the host TreeNode object.

tfamily \textcolor{red}\textless type(abundances)\textgreater \ hotHaloCoolingAbundances() Get the hotHaloCoolingAbundances property of the HotHalo component.

tfamily \textcolor{red}\textless type(varying\_string)();\textgreater \ hotHaloCoolingAbundancesAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \intzero hotHaloCoolingAbundancesCount() Compute the count of evolvable quantities in the hotHaloCoolingAbundances property of the HotHaloStandard component.

tfamily \textcolor{red}\textless double\textgreater \ hotHaloCoolingAngularMomentum() Get the hotHaloCoolingAngularMomentum property of the HotHalo component.

tfamily \textcolor{red}\textless double\textgreater \ hotHaloCoolingAngularMomentumIsGettable() Get the hotHaloCoolingAngularMomentum property of the HotHalo component.

tfamily \logicalzero hotHaloCoolingAngularMomentumIsSettable() Specify whether the hotHaloCoolingAngularMomentum property of the HotHalo component is settable.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the HotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAngularMomentum property of the HotHalo component.

tfamily \logicalzero hotHaloCoolingAngularMomentumRateIsAttached() Return whether the rate method of the hotHaloCoolingAngularMomentum property of the HotHalo component has been attached to a function.

558
16.5. Objects

tfamily \logicalzero hotHaloCoolingAngularMomentumRateIsAttached() Return whether the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component has been attached to a function.

tfamily \textcolor{red}{\textless \textless \textgreater} hotHaloCoolingMass() Get the hotHaloCoolingMass property of the hotHalo component.

tfamily \textcolor{red}{\textless \textless \textgreater} hotHaloCoolingMassAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the hotHaloCoolingMass property of the hotHalo component.

tfamily \intzero hotHaloCoolingMassCount() Compute the count of evolvable quantities in the hotHaloCoolingMass property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless \textless \textgreater} hotHaloCoolingMassIsGettable() Get the hotHaloCoolingMass property of the hotHalo component.

tfamily \logicalzero hotHaloCoolingMassIsSettable() Specify whether the hotHaloCoolingMass property of the hotHalo component is settable.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void hotHaloCoolingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingMass property of the hotHalo component.

tfamily \logicalzero hotHaloCoolingMassRateIsAttached() Return whether the rate method of the hotHaloCoolingMass property of the hotHalo component has been attached to a function.

tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}{\textless \logical \textgreater} isInitialized() Get the isInitialized property of the hotHalo component.

tfamily \textcolor{red}{\textless \textless \textgreater} isInitializedAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the isInitialized property of the hotHalo component.

tfamily \textcolor{red}{\textless \logical \textgreater} isInitializedIsGettable() Get the isInitialized property of the hotHalo component.

tfamily \logicalzero isInitializedIsSettable() Specify whether the isInitialized property of the hotHalo component is settable.

tfamily \void isInitializedSet(<logical> value) Set the isInitialized property of the hotHalo component.

tfamily \textcolor{red}{\textless \textless \textgreater} mass() Get the mass property of the hotHalo component.

tfamily \textcolor{red}{\textless \textless \textgreater} massAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the mass property of the hotHalo component.
16. Coding Galacticus

tfamily \textcolor{red}{\textless double\textgreater} massCold() Get the massCold property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} massColdAttributeMatch(<logical>[requireGettable] → , <logical>[requireSettable] → , <logical>[requireEvolvable] → ) Return a list of implementations that provide the given list off attributes for the massCold property of the hotHalo component.

tfamily \intzero massColdCount() Compute the count of evolvable quantities in the massCold property of the HotHaloColdMode component.

tfamily \textcolor{red}{\textless double\textgreater} massColdIsGettable() Get the massCold property of the hotHalo component.

tfamily \logicalzero massColdIsSettable() Specify whether the massCold property of the hotHalo component is settable.

tfamily \void massColdRate(<double> value) Cumulate to the rate of the massCold property of the HotHaloColdMode component.

tfamily \void massColdSet(<double> value) Set the massCold property of the hotHalo component.

tfamily \intzero massCount() Compute the count of evolvable quantities in the mass property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} massIsGettable() Get the mass property of the hotHalo component.

tfamily \logicalzero massIsSettable() Specify whether the mass property of the hotHalo component is settable.

tfamily \void massRate(<double> value) Cumulate to the rate of the mass property of the HotHaloStandard component.

tfamily \void massScale(<double> value) Set the scale of the mass property of the HotHaloStandard component.

tfamily \void massSet(<double> value) Set the mass property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} massSink() Get the massSink property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} massSinkAttributeMatch(<logical>[requireGettable] → , <logical>[requireSettable] → , <logical>[requireEvolvable] → ) Return a list of implementations that provide the given list off attributes for the massSink property of the hotHalo component.

tfamily \intzero massSinkCount() Compute the count of evolvable quantities in the massSink property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} massSinkIsGettable() Get the massSink property of the hotHalo component.

tfamily \logicalzero massSinkIsSettable() Specify whether the massSink property of the hotHalo component is settable.
16.5. Objects

tfamily \void massSinkRate(<double> value) Cumulate to the rate of the massSink property of the HotHaloStandard component.

tfamily \void nameFromIndex(<integer> count →, <varying_string> name ←) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the hotHalo component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \textcolor{red}\textless double\textgreater outerRadius() Get the outerRadius property of the hotHalo component.

tfamily \textcolor{red}\textless type(varying\_string)\textgreater outerRadiusAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the outerRadius property of the hotHalo component.

tfamily \intzero outerRadiusCount() Compute the count of evolvable quantities in the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusFunction(<function()> deferredFunction) Set the function to be used for the get method of the outerRadius property of the HotHaloVerySimple component.

tfamily \logicalzero outerRadiusIsAttached() Return whether the get method of the outerRadius property of the HotHaloVerySimple component has been attached to a function.

tfamily \textcolor{red}\textless double\textgreater outerRadiusIsGettable() Get the outerRadius property of the hotHalo component.

tfamily \logicalzero outerRadiusIsSettable() Specify whether the outerRadius property of the hotHalo component is settable.

tfamily \void outerRadiusRate(<double> value) Cumulate to the rate of the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusScale(<double> value) Set the scale of the outerRadius property of the HotHaloStandard component.

tfamily \void outerRadiusSet(<double> value) Set the outerRadius property of the hotHalo component.

tfamily \textcolor{red}\textless type(abundances)\textgreater outflowedAbundances() Get the outflowedAbundances property of the hotHalo component.

tfamily \textcolor{red}\textless type(varying\_string)\textgreater outflowedAbundancesAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the outflowedAbundances property of the hotHalo component.

tfamily \intzero outflowedAbundancesCount() Compute the count of evolvable quantities in the outflowedAbundances property of the HotHaloStandard component.
16. Coding Galacticus

tfamily \textcolor{red}{\textless \textless \textless \textless \textless} outflowedAbundancesIsGettable() Get the \textcolor{red}{\textless \textless \textless \textless \textless} outflowedAbundances property of the \textcolor{red}{\textless \textless \textless \textless \textless} hotHalo component.

tfamily \logicalzero outflowedAbundancesIsSettable() Specify whether the \textcolor{red}{\textless \textless \textless \textless \textless} outflowedAbundances property of the \textcolor{red}{\textless \textless \textless \textless \textless} hotHalo component is settable.

tfamily \void outflowedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the \textcolor{red}{\textless \textless \textless \textless \textless} outflowedAbundances property of the \textcolor{red}{\textless \textless \textless \textless \textless} HotHaloStandard component.

tfamily \void outflowedAbundancesScale(<type(abundances)> value) Set the scale of the \textcolor{red}{\textless \textless \textless \textless \textless} outflowedAbundances property of the \textcolor{red}{\textless \textless \textless \textless \textless} HotHaloStandard component.

tfamily \void outflowedAbundancesSet(<type(abundances)> value) Set the \textcolor{red}{\textless \textless \textless \textless \textless} outflowedAbundances property of the \textcolor{red}{\textless \textless \textless \textless \textless} hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} outflowedAngularMomentum() Get the \textcolor{red}{\textless double\textgreater} outflowedAngularMomentum property of the \textcolor{red}{\textless double\textgreater} hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} outflowedAngularMomentumAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the \textcolor{red}{\textless double\textgreater} outflowedAngularMomentum property of the \textcolor{red}{\textless double\textgreater} hotHalo component.

tfamily \intzero outflowedAngularMomentumCount() Compute the count of evolvable quantities in the \textcolor{red}{\textless double\textgreater} outflowedAngularMomentum property of the \textcolor{red}{\textless double\textgreater} HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} outflowedAngularMomentumIsGettable() Get the \textcolor{red}{\textless double\textgreater} outflowedAngularMomentum property of the \textcolor{red}{\textless double\textgreater} hotHalo component.

tfamily \logicalzero outflowedAngularMomentumIsSettable() Specify whether the \textcolor{red}{\textless double\textgreater} outflowedAngularMomentum property of the \textcolor{red}{\textless double\textgreater} hotHalo component is settable.

tfamily \void outflowedAngularMomentumRate(<double> value) Cumulate to the rate of the \textcolor{red}{\textless double\textgreater} outflowedAngularMomentum property of the \textcolor{red}{\textless double\textgreater} HotHaloStandard component.

tfamily \void outflowedAngularMomentumScale(<double> value) Set the scale of the \textcolor{red}{\textless double\textgreater} outflowedAngularMomentum property of the \textcolor{red}{\textless double\textgreater} HotHaloStandard component.

tfamily \void outflowedAngularMomentumSet(<double> value) Set the \textcolor{red}{\textless double\textgreater} outflowedAngularMomentum property of the \textcolor{red}{\textless double\textgreater} hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} outflowedMass() Get the \textcolor{red}{\textless double\textgreater} outflowedMass property of the \textcolor{red}{\textless double\textgreater} hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} outflowedMassAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the \textcolor{red}{\textless double\textgreater} outflowedMass property of the \textcolor{red}{\textless double\textgreater} hotHalo component.

tfamily \intzero outflowedMassCount() Compute the count of evolvable quantities in the \textcolor{red}{\textless double\textgreater} outflowedMass property of the \textcolor{red}{\textless double\textgreater} HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} outflowedMassIsGettable() Get the \textcolor{red}{\textless double\textgreater} outflowedMass property of the \textcolor{red}{\textless double\textgreater} hotHalo component.

tfamily \logicalzero outflowedMassIsSettable() Specify whether the \textcolor{red}{\textless double\textgreater} outflowedMass property of the \textcolor{red}{\textless double\textgreater} hotHalo component is settable.
16.5. Objects

tfamily \void outflowedMassRate(<double> value) Cumulate to the rate of the outflowedMass property of the HotHaloStandard component.

tfamily \void outflowedMassScale(<double> value) Set the scale of the outflowedMass property of the HotHaloStandard component.

tfamily \void outflowedMassSet(<double> value) Set the outflowedMass property of the hotHalo component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} outflowingAbundances() Get the outflowingAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless varying\_string\textgreater\textgreater} outflowingAbundancesAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the outflowingAbundances property of the hotHalo component

tfamily \intzero outflowingAbundancesCount() Compute the count of evolvable quantities in the outflowingAbundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} outflowingAbundancesIsGettable() Get the outflowingAbundances property of the hotHalo component.

tfamily \logicalzero outflowingAbundancesIsSettable() Specify whether the outflowingAbundances property of the hotHalo component is settable.

tfamily \void outflowingAbundancesRate(<type(abundances) > value) Cumulate to the rate of the outflowingAbundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} outflowingAngularMomentum() Get the outflowingAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless varying\_string\textgreater\textgreater} outflowingAngularMomentumAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the outflowingAngularMomentum property of the hotHalo component

tfamily \intzero outflowingAngularMomentumCount() Compute the count of evolvable quantities in the outflowingAngularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} outflowingAngularMomentumIsGettable() Get the outflowingAngularMomentum property of the hotHalo component.

tfamily \logicalzero outflowingAngularMomentumIsSettable() Specify whether the outflowingAngularMomentum property of the hotHalo component is settable.

tfamily \void outflowingAngularMomentumRate(<double> value) Cumulate to the rate of the outflowingAngularMomentum property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} outflowingMass() Get the outflowingMass property of the hotHalo component.

tfamily \textcolor{red}{\textless varying\_string\textgreater\textgreater} outflowingMassAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the outflowingMass property of the hotHalo component.
16. Coding GALACTICUS

tfamily \intzero outflowingMassCount() Compute the count of evolvable quantities in the outflowingMass property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} outflowingMassIsGettable() Get the outflowingMass property of the hotHalo component.

tfamily \logicalzero outflowingMassIsSettable() Specify whether the outflowingMass property of the hotHalo component is settable.

tfamily \void outflowingMassRate(<double> value) Cumulate to the rate of the outflowingMass property of the hotHalo component.

tfamily \void outflowingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingMass property of the HotHaloVerySimple component.

tfamily \logicalzero outflowingMassRateIsAttached() Return whether the rate method of the outflowingMass property of the HotHaloVerySimple component has been attached to a function.

tfamily \logicalzero outflowTrackingIsActive() Return whether the outflowTracking implementation of the hotHalo component class is active.

tfamily \void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer> doubleProperty↔, <integer> doubleBufferCount↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \void outputCount(<integer>integerPropertyCount↔, <integer>doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \void outputNames(<integer>integerProperty↔, <char[*](:)>integerPropertyNames↔, <char[*](:)>integerPropertyComments↔, <double(:)>integerPropertyUnitsSI↔, <integer>doubleProperty↔, <char[*](:)>doublePropertyNames↔, <char[*](:)>doublePropertyComments↔, <double(:)>doublePropertyUnitsSI↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \doublezero potential(<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

tfamily \void readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezero rotationCurve(<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the hotHalo component class is active.
16.5. Objects

tfamily \textcolor{red}{\textless type(abundances)\textgreater} strippedAbundances() Get the strippedAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} strippedAbundancesAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list off attributes for the strippedAbundances property of the hotHalo component.

tfamily \intzero strippedAbundancesCount() Compute the count of evolvable quantities in the strippedAbundances property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} strippedAbundancesIsGettable() Get the strippedAbundances property of the hotHalo component.

tfamily \logicalzero strippedAbundancesIsSettable() Specify whether the strippedAbundances property of the hotHalo component is settable.

tfamily \void strippedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the strippedAbundances property of the HotHaloStandard component.

tfamily \void strippedAbundancesScale(<type(abundances)> value) Set the scale of the strippedAbundances property of the HotHaloStandard component.

tfamily \void strippedAbundancesSet(<type(abundances)> value) Set the strippedAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} strippedMass() Get the strippedMass property of the hotHalo component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} strippedMassAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list off attributes for the strippedMass property of the hotHalo component.

tfamily \intzero strippedMassCount() Compute the count of evolvable quantities in the strippedMass property of the HotHaloStandard component.

tfamily \textcolor{red}{\textless double\textgreater} strippedMassIsGettable() Get the strippedMass property of the hotHalo component.

tfamily \logicalzero strippedMassIsSettable() Specify whether the strippedMass property of the hotHalo component is settable.

tfamily \void strippedMassRate(<double> value) Cumulate to the rate of the strippedMass property of the HotHaloStandard component.

tfamily \void strippedMassScale(<double> value) Set the scale of the strippedMass property of the HotHaloStandard component.

tfamily \void strippedMassSet(<double> value) Set the strippedMass property of the hotHalo component.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical \rightarrow, <componentType>[componentType] \rightarrow, <massType>[massType] \rightarrow, <weightBy>[weightBy] \rightarrow, <integer>[weightIndex] \rightarrow, <logical>[haloLoaded] \rightarrow) Compute the surface density.
16. Coding GALACTICUS

tfamily \textcolor{red}{\textless type(abundances)\textgreater} trackedOutflowAbundances() Get the trackedOutflowAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string):(:)\textgreater} trackedOutflowAbundancesAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the trackedOutflowAbundances property of the hotHalo component.

tfamily \intzero trackedOutflowAbundancesCount() Compute the count of evolvable quantities in the trackedOutflowAbundances property of the HotHaloOutflowTracking component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} trackedOutflowAbundancesIsGettable() Get the trackedOutflowAbundances property of the hotHalo component.

tfamily \logicalzero trackedOutflowAbundancesIsSettable() Specify whether the trackedOutflowAbundances property of the hotHalo component is settable.

tfamily \void trackedOutflowAbundancesRate(<type(abundances)> value) Cumulate to the rate of the trackedOutflowAbundances property of the HotHaloOutflowTracking component.

tfamily \void trackedOutflowAbundancesScale(<type(abundances)>value) Set the scale of the trackedOutflowAbundances property of the HotHaloOutflowTracking component.

tfamily \void trackedOutflowAbundancesSet(<type(abundances)>value) Set the trackedOutflowAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} trackedOutflowMass() Get the trackedOutflowMass property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string):(:)\textgreater} trackedOutflowMassAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the trackedOutflowMass property of the hotHalo component.

tfamily \intzero trackedOutflowMassCount() Compute the count of evolvable quantities in the trackedOutflowMass property of the HotHaloOutflowTracking component.

tfamily \textcolor{red}{\textless double\textgreater} trackedOutflowMassIsGettable() Get the trackedOutflowMass property of the hotHalo component.

tfamily \logicalzero trackedOutflowMassIsSettable() Specify whether the trackedOutflowMass property of the hotHalo component is settable.

tfamily \void trackedOutflowMassRate(<double> value) Cumulate to the rate of the trackedOutflowMass property of the HotHaloOutflowTracking component.

tfamily \void trackedOutflowMassScale(<double> value) Set the scale of the trackedOutflowMass property of the HotHaloOutflowTracking component.

tfamily \void trackedOutflowMassSet(<double> value) Set the trackedOutflowMass property of the hotHalo component.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.
16.5. Objects

tfamily \textcolor{red}{\textless \textless \textgreater}\text{unaccretedMass() Get the unaccretedMass property of the hotHalo component.}

tfamily \textcolor{red}{\textless \textless \textgreater}\text{unaccretedMassAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow)} Return a list of implementations that provide the given list of attributes for the unaccretedMass property of the hotHalo component.

tfamily \textcolor{red}{\textless \textless \textgreater}\text{unaccretedMassCount() Compute the count of evolvable quantities in the unaccretedMass property of the HotHaloStandard component.}

tfamily \textcolor{red}{\textless \textless \textgreater}\text{unaccretedMassIsGettable() Get the unaccretedMass property of the hotHalo component.}

tfamily \textcolor{red}{\textless \textless \textgreater}\text{unaccretedMassIsSettable() Specify whether the unaccretedMass property of the hotHalo component is settable.}

tfamily \textcolor{red}{\textless \textless \textgreater}\text{unaccretedMassRate(<double> value) Cumulate to the rate of the unaccretedMass property of the HotHaloStandard component.}

tfamily \textcolor{red}{\textless \textless \textgreater}\text{unaccretedMassScale(<double> value) Set the scale of the unaccretedMass property of the HotHaloStandard component.}

tfamily \textcolor{red}{\textless \textless \textgreater}\text{unaccretedMassSet(<double> value) Set the unaccretedMass property of the hotHalo component.}

**tfamily nodeComponentIndices**

tfamily \textcolor{red}{\textless \textless \textgreater}\text{branchTip() Get the branchTip property of the indices component.}

tfamily \textcolor{red}{\textless \textless \textgreater}\text{branchTipAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow)} Return a list of implementations that provide the given list of attributes for the branchTip property of the indices component.

**tfamily nodeComponentIndices**

tfamily \textcolor{red}{\textless \textless \textgreater}\text{branchTipIsGettable() Get the branchTip property of the indices component.}

**tfamily nodeComponentIndices**

tfamily \textcolor{red}{\textless \textless \textgreater}\text{branchTipIsSettable() Specify whether the branchTip property of the indices component is settable.}

**tfamily nodeComponentIndices**

tfamily \textcolor{red}{\textless \textless \textgreater}\text{branchTipSet(<integer(kind=kind_int8)> value) Set the branchTip property of the indices component.}

**tfamily nodeComponentIndices**

tfamily \textcolor{red}{\textless \textless \textgreater}\text{builder(<\text{*type(node)>componentDefinition}) Build a nodeComponent from a supplied XML definition.

**tfamily nodeComponentIndices**

tfamily \textcolor{red}{\textless \textless \textgreater}\text{density(<double(3)> positionSpherical \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <weightBy> [weightBy] \rightarrow, <integer> [weightIndex] \rightarrow, <logical> [haloLoaded] \rightarrow)} Compute the density.

**tfamily nodeComponentIndices**

tfamily \textcolor{red}{\textless \textless \textgreater}\text{deserializeRates(<double(:)> array \leftarrow)} Deserialize the evolvable rates from an array.

**tfamily nodeComponentIndices**

tfamily \textcolor{red}{\textless \textless \textgreater}\text{deserializeScales(<double(:)> array \leftarrow)} Deserialize the evolvable scales from an array.
tfamily \void deserializeValues(<double(:)> array→) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \double zeroEnclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)
Compute the mass enclosed within a radius.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count→, <varying_string> name←) Return the name of a property given is index.

tfamily \logical zero nullIsActive() Return whether the null implementation of the indices component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty↔, <char[*](::)> integerPropertyNames↔, <char[*](::)> integerPropertyComments↔, <double(:,:)> integerPropertyUnitsSI↔, <integer> doubleProperty↔, <char[*](::)> doublePropertyNames↔, <char[*](::)> doublePropertyComments↔, <double(:,:)> doublePropertyUnitsSI↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \double zero potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

tfamily \double zero rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.
16.5. Objects

tfamily `doublezero rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →)` Compute the rotation curve gradient.

tfamily `intzero serializeCount()` Return a count of the number of evolvable quantities to be evolved.

tfamily `void serializeRates(<double(:)> array →)` Serialize the evolvable rates to an array.

tfamily `void serializeScales(<double(:)> array →)` Serialize the evolvable scales to an array.

tfamily `void serializeValues(<double(:)> array →)` Serialize the evolvable quantities to an array.

tfamily `logicalzero standardIsActive()` Return whether the standard implementation of the indices component class is active.

tfamily `doublezero surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →)` Compute the surface density.

tfamily `textcolor{red}{textless type(varying \_string)\textgreater} type()` Return the type of this object.

**tfamily nodeComponentIndicesNull**

tfamily `textcolor{red}{textless integer(kind=kind\_int8)\textgreater} branchTip()` Get the branchTip property of the indices component.

tfamily `textcolor{red}{textless type(varying \_string)\textgreater} branchTipAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →)` Return a list of implementations that provide the given list of attributes for the branchTip property of the indices component.

tfamily `textcolor{red}{textless integer(kind=kind\_int8)\textgreater} branchTipIsGettable()` Get the branchTip property of the indices component.

tfamily `logicalzero branchTipIsSettable()` Specify whether the branchTip property of the indices component is settable.

tfamily `void branchTipSet(<integer(kind=kind\_int8)> value)` Set the branchTip property of the indices component.

tfamily `void builder(<\^\text{type(node)}\textgreater\text{componentDefinition} →)` Build a nodeComponent from a supplied XML definition.

tfamily `doublezero density(<double(3)> positionSpherical →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →)` Compute the density.

tfamily `void deserializeRates(<double(:)> array ←)` Deserialize the evolvable rates from an array.

tfamily `void deserializeScales(<double(:)> array ←)` Deserialize the evolvable scales from an array.

tfamily `void deserializeValues(<double(:)> array ←)` Deserialize the evolvable quantities from an array.

tfamily `void destroy()` Destroy the object.

tfamily `void dump()` Generate an ASCII dump of all properties.

tfamily `void dumpRaw(<integer> fileHandle →)` Generate a binary dump of all properties.
tfamily \ void dumpXML() Generate an XML dump of all properties.

tfamily \ double zero enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \*

<table>
<thead>
<tr>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>{textcolor{red}}</td>
</tr>
</tbody>
</table>
| \ textless *
| type(treeNode) \ textgreater |
| host() Return a pointer to the host treeNode object. |

tfamily \ void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \ void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \ void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \ void initialize() Initialize the object.

tfamily \ void nameFromIndex(<integer> count→, <varying_string> name←) Return the name of a property given is index.

tfamily \ logical zero nullIsActive() Return whether the null implementation of the indices component class is active.

tfamily \ void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \ void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \ void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \ void outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

<table>
<thead>
<tr>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>\ void outputNames(&lt;integer&gt; integerProperty↔, &lt;char<a href="">*</a>&gt; integerPropertyNames↔, &lt;char<a href="">*</a>&gt; integerPropertyComments↔, &lt;double(:)&gt; integerPropertyUnitsSI↔, &lt;integer&gt; doubleProperty↔, &lt;char<a href="">*</a>&gt; doublePropertyNames↔, &lt;char<a href="">*</a>&gt; doublePropertyComments↔, &lt;double(:)&gt; doublePropertyUnitsSI↔, &lt;double&gt; time→, &lt;integer&gt; instance→) Generate names of outputtable properties.</td>
</tr>
</tbody>
</table>

tfamily \ double zero potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

tfamily \ void readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \ double zero rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

tfamily \ double zero rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

tfamily \ int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.
16.5. Objects

tfamily \void serializeRates(<double();> array→) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double();> array→) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double();> array→) Serialize the evolvable quantities to an array.

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the indices component class is active.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

tfamily nodeComponentIndicesStandard


tfamily \textcolor{red}{\textless integer(kind=kind\_int8)\textgreater} branchTip() Get the branchTip property of the indices component.

tfamily \textcolor{red}{\textless textless type(varying\_string)\textgreater} branchTipAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the branchTip property of the indices component.

tfamily \textcolor{red}{\textless integer(kind=kind\_int8)\textgreater} branchTipIsGettable() Get the branchTip property of the indices component.

tfamily \logicalzero branchTipIsSettable() Specify whether the branchTip property of the indices component is settable.

tfamily \void branchTipSet(<integer(kind=kind\_int8)> value) Set the branchTip property of the indices component.

tfamily \void builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily \doublezero density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double();> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double();> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double();> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.
16. Coding GALACTICUS

tfamily \doublezero enclosedMass(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <weightBy> [weightBy] \rightarrow, <integer> [weightIndex] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host \textit{treeNode} object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the \textit{hotHaloCoolingAbundances} property of the \textit{hotHalo} component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the \textit{hotHaloCoolingAngularMomentum} property of the \textit{hotHalo} component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the \textit{hotHaloCoolingMass} property of the \textit{hotHalo} component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count \rightarrow, <varying_string> name \leftarrow) Return the name of a property given its index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the indices component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty \leftrightarrow, <integer> integerBufferCount \leftrightarrow, <integer(:,:)> integerBuffer \leftrightarrow, <integer> doubleProperty \leftrightarrow, <integer> doubleBufferCount \leftrightarrow, <double(:,:)> doubleBuffer \leftrightarrow, <double> time \rightarrow, <integer> instance \rightarrow) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount \leftrightarrow, <integer> doublePropertyCount \leftrightarrow, <double> time \rightarrow, <integer> instance \rightarrow) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty \leftrightarrow, <char[*](:)> integerPropertyNames \leftrightarrow, <char[*](:)> integerPropertyComments \leftrightarrow, <double(:)> integerPropertyUnitsSI \leftrightarrow, <integer> doubleProperty \leftrightarrow, <char[*](:)> doublePropertyNames \leftrightarrow, <char[*](:)> doublePropertyComments \leftrightarrow, <double(:)> doublePropertyUnitsSI \leftrightarrow, <double> time \rightarrow, <integer> instance \rightarrow) Generate names of outputtable properties.

tfamily \doublezero potential(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the gravitational potential.

tfamily \void readRaw(<integer> fileHandle \leftrightarrow) Read a binary dump of the \textit{nodeComponent} from the given \texttt{fileHandle}.

tfamily \doublezero rotationCurve(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array \rightarrow) Serialize the evolvable rates to an array.
16.5. Objects

tfamily void serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

tfamily void serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

tfamily logical zero standardIsActive() Return whether the standard implementation of the indices component class is active.

tfamily double zero surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the surface density.

tfamily \textcolor{red}{\textless} type(varying\_string)\textgreater type() Return the type of this object.

tfamily nodeComponentInterOutput
tfamily void builder(<*type(node)> componentDefinition →) Build a nodeComponent from a supplied XML definition.

tfamily double zero density(<double(3)> positionSpherical →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the density.

tfamily void deserializeRates(<double(:)> array ←) Deserialize the evolvable rates from an array.

tfamily void deserializeScales(<double(:)> array ←) Deserialize the evolvable scales from an array.

tfamily void deserializeValues(<double(:)> array ←) Deserialize the evolvable quantities from an array.

tfamily void destroy() Destroy the object.

tfamily \textcolor{red}{\textless} double\textgreater diskStarFormationRate() Get the diskStarFormationRate property of the interOutput component.

tfamily \textcolor{red}{\textless} type(varying\_string)(:)\textgreater diskStarFormationRateAttributeMatcher(<logical> [requireGettable] →, <logical> [requireSettable] →, <logical> [requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the diskStarFormationRate property of the interOutput component

tfamily int zero diskStarFormationRateCount() Compute the count of evolvable quantities in the diskStarFormationRate property of the InterOutputStandard component.

tfamily \textcolor{red}{\textless} double\textgreater diskStarFormationRateIsGettable() Get the diskStarFormationRateIsGettable() Get the diskStarFormationRate property of the interOutput component.

tfamily logical zero diskStarFormationRateIsSettable() Specify whether the diskStarFormationRate property of the interOutput component is settable.

tfamily void diskStarFormationRateRate(<double> value) Cumulate to the rate of the diskStarFormationRate property of the InterOutputStandard component.

tfamily void diskStarFormationRateScale(<double> value) Set the scale of the diskStarFormationRate property of the InterOutputStandard component.

tfamily void diskStarFormationRateSet(<double> value) Set the diskStarFormationRate property of the interOutput component.
tfamily \ void dump() Generate an ASCII dump of all properties.

tfamily \ void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \ void dumpXML() Generate an XML dump of all properties.

tfamily \ doublezero enclosedMass(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \ textcolor{red}\textless *type(treeNode)\textgreater host() Return a pointer to the host treeNode object.

tfamily \ void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \ void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \ void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \ void initialize() Initialize the object.

tfamily \ void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

tfamily \ logicalzero nullIsActive() Return whether the null implementation of the interOutput component class is active.

tfamily \ void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \ void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \ void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)>integerBuffer↔, <integer:doubleProperty↔, <integer:doubleBufferCount↔, <double(:,:)>doubleBuffer↔, <double>time→, <integer>instance→) Generate values of outputtable properties.

tfamily \ void outputCount(<integer>integerPropertyCount↔, <integer>doublePropertyCount↔, <double>time→, <integer>instance→) Compute a count of outputtable properties.

tfamily \ void outputNames(<integer>integerProperty↔, <char[*]>integerPropertyNames↔, <char[*]>integerPropertyComments↔, <double:integerPropertyUnitsSI↔, <integer:doubleProperty↔, <char[*]>doublePropertyNames↔, <char[*]>doublePropertyComments↔, <double:doublePropertyUnitsSI↔, <double>time→, <integer>instance→) Generate names of outputtable properties.

tfamily \ doublezero potential(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the gravitational potential.

tfamily \ doublezero rotationCurve(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the rotation curve.

tfamily \ doublezero rotationCurveGradient(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the rotation curve gradient.
16.5. Objects

tfamily intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily void serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

tfamily void serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

tfamily void serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

tfamily textcolor{red}{\textless double\textgreater}spheroidStarFormationRate() Get the spheroidStarFormationRate property of the interOutput component.

tfamily textcolor{red}{\textless type(varying\_string):(\textgreater)}spheroidStarFormationRateAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the spheroidStarFormationRate property of the interOutput component.

tfamily intzero spheroidStarFormationRateCount() Compute the count of evolvable quantities in the spheroidStarFormationRate property of the InterOutputStandard component.

tfamily textcolor{red}{\textless double\textgreater}spheroidStarFormationRateIsGettable() Get the spheroidStarFormationRate property of the interOutput component.

tfamily logicalzerospheroidStarFormationRateIsSettable() Specify whether the spheroidStarFormationRate property of the interOutput component is settable.

tfamily void spheroidStarFormationRateRate(<double> value) Cumulatethe rate of the spheroidStarFormationRate property of the InterOutputStandard component.

tfamily void spheroidStarFormationRateScale(<double> value) Set the scale of the spheroidStarFormationRate property of the InterOutputStandard component.

tfamily void spheroidStarFormationRateSet(<double> value) Set the spheroidStarFormationRate property of the interOutput component.

tfamily logicalzerostandardIsActive() Return whether the standard implementation of the interOutput component class is active.

tfamily doublezerosurfaceDensity(<double(3)> positionCylindrical→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the surface density.

tfamily textcolor{red}{\textless type(varying\_string):(\textgreater)} type() Return the type of this object.

tfamily nodeComponentInterOutputNull

tfamily void builder(<"type(node)"componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily doublezerosurfaceDensity(<double(3)> positionSpherical→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the density.

tfamily void deserializerrates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily void deserializerratesScales(<double(:)> array←) Deserialize the evolvable scales from an array.
tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \textcolor{red}{\textless double\textgreater} diskStarFormationRate() Get the diskStarFormationRate property of the interOutput component.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.
16.5. Objects

tfamily \logicalzero nullIsActive() Return whether the null implementation of the interOutput component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

tfamily \void outputCount(<integer>integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily \void outputNames(<integer>integerProperty ↔, <char[*](:)>integerPropertyNames ↔, <char[*](:)> integerPropertyComments ↔, <double(:)>integerPropertyUnitsSI ↔, <integer>doubleProperty ↔, <char[*](:)>doublePropertyNames ↔, <char[*](:)> doublePropertyComments ↔, <double(:)> doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

tfamily \doublezeropotential(<double>radius →, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the gravitational potential.

tfamily \void readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezerotationCurve(<double>radius →, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the rotation curve.

tfamily \doublezerotationCurveGradient(<double>radius →, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

tfamily \textcolor{red}{\textless double\textgreater}spheroidStarFormationRate() Get the spheroidStarFormationRate property of the interOutput component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater spheroidStarFormationRateAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the spheroidStarFormationRate property of the interOutput component.

tfamily \intzero spheroidStarFormationRateCount() Compute the count of evolvable quantities in the spheroidStarFormationRate property of the InterOutputStandard component.

tfamily \textcolor{red}{\textless double\textgreater}spheroidStarFormationRateIsGettable() Get the spheroidStarFormationRate property of the interOutput component.

tfamily \logicalzero spheroidStarFormationRateIsSettable() Specify whether the spheroidStarFormationRate property of the interOutput component is settable.
tfamily \void spheroidStarFormationRateRate(<double> value) Cumulate to the rate of the spheroidStarFormationRate property of the InterOutputStandard component.

tfamily \void spheroidStarFormationRateScale(<double> value) Set the scale of the spheroidStarFormationRate property of the InterOutputStandard component.

tfamily \void spheroidStarFormationRateSet(<double> value) Set the spheroidStarFormationRate property of the interOutput component.

tfamily \logical zero standardIsActive() Return whether the standard implementation of the interOutput component class is active.

tfamily \double zero surfaceDensity(<double(3)> positionCylindrical \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <weightBy> [weightBy] \rightarrow, <integer> [weightIndex] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the surface density.

tfamily \textcolor{red}{}\textless type(varying \_string)\textgreater\ type() Return the type of this object.

tfamily nodeComponentInterOutputStandard

tfamily \void builder(<\*type(node)> componentDefinition \rightarrow) Build a nodeComponent from a supplied XML definition.

tfamily \double zero density(<double(3)> positionSpherical \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <weightBy> [weightBy] \rightarrow, <integer> [weightIndex] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the density.

tfamily \void deserializeRates(<double(:) array ←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:) array ←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:) array ←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \textcolor{red}{}\textless double\textgreater\ diskStarFormationRate() Get the diskStarFormationRate property of the interOutput component.

tfamily \textcolor{red}{}\textless type(varying \_string)(:)\textgreater\ diskStarFormationRateAttributeMatch(<logical> [requireGettable] \rightarrow, <logical> [requireSettable] \rightarrow, <logical> [requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the diskStarFormationRate property of the interOutput component

tfamily \int zero diskStarFormationRateCount() Compute the count of evolvable quantities in the diskStarFormationRate property of the InterOutputStandard component.

tfamily \textcolor{red}{}\textless double\textgreater\ diskStarFormationRateIsGettable() Get the diskStarFormationRate property of the interOutput component.

tfamily \logical zero diskStarFormationRateIsSettable() Specify whether the diskStarFormationRate property of the interOutput component is settable.

tfamily \void diskStarFormationRateRate(<double> value) Cumulate to the rate of the diskStarFormationRate property of the InterOutputStandard component.
tfamily \void diskStarFormationRateScale(<double> value) Set the scale of the diskStarFormationRate property of the InterOutputStandard component.

tfamily \void diskStarFormationRateSet(<double> value) Set the diskStarFormationRate property of the interOutput component.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \double zeroenclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host TreeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

tfamily \logical zero nullIsActive() Return whether the null implementation of the interOutput component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \void outputCount(<integer>integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \void outputNames(<integer>integerProperty↔, <char[^]*[;]> integerPropertyNames↔, <char[^]*[;]> integerPropertyComments↔, <double> integerPropertyUnitsSI↔, <integer> doubleProperty↔, <char[^]*[;]> doublePropertyNames↔, <char[^]*[;]> doublePropertyComments↔, <double(:,:)> doublePropertyUnitsSI↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \double zero potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.
16. **Coding GALACTICUS**

tfamily \void readRaw(<integer> fileHandle \rightarrow) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezero rotationCurve(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:) > array \rightarrow) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:) > array \rightarrow) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:) > array \rightarrow) Serialize the evolvable quantities to an array.

tfamily \textcolor{red}{<double\textgreater}spheroidStarFormationRate() Get the spheroidStarFormationRate property of the interOutput component.

tfamily \textcolor{red}{< type(varying_string):(>)\textgreater}spheroidStarFormationRateAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the spheroidStarFormationRate property of the interOutput component.

tfamily \intzero spheroidStarFormationRateCount() Compute the count of evolvable quantities in the spheroidStarFormationRate property of the InterOutputStandard component.

tfamily \textcolor{red}{< type(varying_string):>} type() Return the type of this object.

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the interOutput component class is active.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <weightBy> [weightBy] \rightarrow, <integer> [weightIndex] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the surface density.

tfamily \textcolor{red}{< type(varying_string):>} type() Return the type of this object.
tfamily nodeComponentMassFlowStatistics

tfamily \void builder(<\text{node})\text{componentDefinition\rightarrow}) Build a nodeComponent from a supplied XML definition.

tfamily \textcolor{red}{\textless \text{double}\textgreater} cooledMass() Get the cooledMass property of the massFlowStatistics component.

tfamily \textcolor{red}{\textless \text{double}\textgreater} cooledMassAttributeMatch(<\text{logical}[\text{requireGettable}\rightarrow], <\text{logical}[\text{requireSettable}\rightarrow], <\text{logical}[\text{requireEvolvable}\rightarrow]) Return a list of implementations that provide the given list of attributes for the cooledMass property of the massFlowStatistics component.

tfamily \intzero cooledMassCount() Compute the count of evolvable quantities in the cooledMass property of the MassFlowStatisticsStandard component.

tfamily \textcolor{red}{\textless \text{double}\textgreater} cooledMassIsGettable() Get the cooledMass property of the massFlowStatistics component.

tfamily \logicalzero cooledMassIsSettable() Specify whether the cooledMass property of the massFlowStatistics component is settable.

tfamily \void cooledMassRate(<\text{double}> value) Cumulate to the rate of the cooledMass property of the MassFlowStatisticsStandard component.

tfamily \void cooledMassScale(<\text{double}> value) Set the scale of the cooledMass property of the MassFlowStatisticsStandard component.

tfamily \void cooledMassSet(<\text{double}> value) Set the cooledMass property of the massFlowStatistics component.

tfamily \doublezero density(<\text{double}(3)> positionSpherical\rightarrow, <\text{componentType}> [\text{componentType}\rightarrow], <\text{massType}> [\text{massType}\rightarrow], <\text{weightBy}> [\text{weightBy}\rightarrow], <\text{integer}> [\text{weightIndex}\rightarrow], <\text{logical}> [\text{haloLoaded}\rightarrow]) Compute the density.

tfamily \void deserializeRates(<\text{double}(\rightarrow) array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<\text{double}(\rightarrow) array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<\text{double}(\rightarrow) array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<\text{integer}> fileHandle\rightarrow) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<\text{double}> radius\rightarrow, <\text{componentType}> [\text{componentType}\rightarrow], <\text{massType}> [\text{massType}\rightarrow], <\text{weightBy}> [\text{weightBy}\rightarrow], <\text{integer}> [\text{weightIndex}\rightarrow], <\text{logical}> [\text{haloLoaded}\rightarrow]) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless \text{type(treeNode)}\textgreater} host() Return a pointer to the host treeNode object.
tfamily \ void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \ void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \ void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \ void initialize() Initialize the object.

tfamily \ void nameFromIndex(<integer> count →, <varying_string>name←) Return the name of a property given is index.

tfamily \ logicalzero nullIsActive() Return whether the null implementation of the massFlowStatistics component class is active.

tfamily \ void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \ void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \ void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \ void outputCount(<integer>integerPropertyCount ↔, <integer>doublePropertyCount ↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \ void outputNames(<integer>integerProperty ↔, <char[*](:)>integerPropertyNames ↔, <char[*](:)>integerPropertyComments ↔, <double(:)>integerPropertyUnitsSI ↔, <integer>doubleProperty ↔, <char[*](:)>doublePropertyNames ↔, <char[*](:)>doublePropertyComments ↔, <double(:)> doublePropertyUnitsSI ↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \ doublezero potential(<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

tfamily \ doublezero rotationCurve(<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

tfamily \ doublezero rotationCurveGradient(<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

tfamily \ intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \ void serializeRates(<double(:):> array→) Serialize the evolvable rates to an array.

tfamily \ void serializeScales(<double(:):> array→) Serialize the evolvable scales to an array.

tfamily \ void serializeValues(<double(:):> array→) Serialize the evolvable quantities to an array.

tfamily \ logicalzero standardIsActive() Return whether the standard implementation of the massFlowStatistics component class is active.

tfamily \ doublezero surfaceDensity(<double(3)>positionCylindrical→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.
16.5. Objects

tfamily \textcolor{red}{\textless type(varying\_\_string)\textgreater} type() Return the type of this object.

tfamily nodeComponentMassFlowStatisticsNull

tfamily \void builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily \textcolor{red}{\textless double\textgreater} cooledMass() Get the cooledMass property of the massFlowStatistics component.

tfamily cooledMassAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the cooledMass property of the massFlowStatistics component.

tfamily \intzero cooledMassCount() Compute the count of evolvable quantities in the cooledMass property of the MassFlowStatisticsStandard component.

tfamily cooledMassIsGettable() Get the cooledMass property of the massFlowStatistics component.

tfamily cooledMassIsSettable() Specify whether the cooledMass property of the massFlowStatistics component is settable.

tfamily \void cooledMassRate(<double> value) Cumulate to the rate of the cooledMass property of the MassFlowStatisticsStandard component.

tfamily \void cooledMassScale(<double>value) Set the scale of the cooledMass property of the MassFlowStatisticsStandard component.

tfamily \void cooledMassSet(<double>value) Set the cooledMass property of the massFlowStatistics component.

tfamily density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

tfamily deserializedRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily deserializedScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily deserializedValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.
tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host TreeNode object.

tfamily void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily void initialize() Initialize the object.

tfamily void nameFromIndex(<integer> count →, <varying_string>name←) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the massFlowStatistics component class is active.

tfamily void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily void outputCount(<integer>integerPropertyCount ↔, <integer>doublePropertyCount ↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily void outputNames(<integer>integerProperty ↔, <char[*]()> integerPropertyNames ↔, <char[*]()> integerPropertyComments ↔, <double(:)>integerPropertyUnitsSI ↔, <integer>doubleProperty ↔, <char[*]()> doublePropertyNames ↔, <char[*]()> doublePropertyComments ↔, <double(:)> doublePropertyUnitsSI ↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \doublezeropotential(<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the gravitational potential.

tfamily \void readRaw(<integer>fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezero rotationCurve(<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)>array→) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)>array→) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)>array→) Serialize the evolvable quantities to an array.
tfamily \logicalzero standardIsActive() Return whether the standard implementation of the massFlowStatistics component class is active.

tfamily double surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.

tfamily {textcolor{red}{\textless type(varying\_string)\textgreater} type()} Return the type of this object.

tfamily nodeComponentMassFlowStatisticsStandard

tfamily \void builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily {textcolor{red}{\textless double\textgreater} cooledMass()} Get the cooledMass property of the massFlowStatistics component.

tfamily {textcolor{red}{\textless double\textgreater} cooledMassAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→)} Return a list of implementations that provide the given list off attributes for the cooledMass property of the massFlowStatistics component.


tfamily {textcolor{red}{\textless double\textgreater} cooledMassIsGettable()} Get the cooledMass property of the massFlowStatistics component.

tfamily \logicalzero cooledMassIsSettable() Specify whether the cooledMass property of the massFlowStatistics component is settable.

tfamily \void cooledMassRate(<double> value) Cumulate to the rate of the cooledMass property of the MassFlowStatisticsStandard component.

tfamily \void cooledMassScale(<double> value) Set the scale of the cooledMass property of the MassFlowStatisticsStandard component.

tfamily \void cooledMassSet(<double> value) Set the cooledMass property of the massFlowStatistics component.

tfamily double density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.
16. Coding Galacticus

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(\double radius→, \componentType [componentType]→, \massType [massType]→, \weightBy [weightBy]→, \integer [weightIndex]→, \logical [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(\type(abundances) value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(\double value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(\double value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

// Other functions are similar to the above.

tfamily \void output(\integer integerProperty↔, \integer integerBufferCount↔, \integer(;) integerBuffer↔, \integer doubleProperty↔, \integer doubleBufferCount↔, \double(;) doubleBuffer↔, \double time→, \integer instance→) Generate values of outputtable properties.

tfamily \void outputCount(\integer integerPropertyCount↔, \integer doublePropertyCount↔, \double time→, \integer instance→) Compute a count of outputtable properties.

// Other function mappings are similar to the above.

tfamily \doublezero potential(\double radius→, \componentType [componentType]→, \massType [massType]→, \logical [haloLoaded]→) Compute the gravitational potential.

// Other functions are similar to the above.

tfamily \void readRaw(\integer fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

// Other function mappings are similar to the above.

tfamily \doublezero rotationCurve(\double radius→, \componentType [componentType]→, \massType [massType]→, \logical [haloLoaded]→) Compute the rotation curve.

// Other function mappings are similar to the above.

tfamily \doublezero rotationCurveGradient(\double radius→, \componentType [componentType]→, \massType [massType]→, \logical [haloLoaded]→) Compute the rotation curve gradient.

// Other function mappings are similar to the above.

tfamily \integer zero serializeCount() Return a count of the number of evolvable quantities to be evolved.
16.5. Objects

tfamily \void serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

ntfamily \logical zero standardIsActive() Return whether the standard implementation of the massFlow-
Statistics component class is active.

tfamily \double zero surfaceDensity(<double(3)> positionCylindrical→, <componentType> [compone-
ntypeType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→,
<logical> [haloLoaded]→) Compute the surface density.

ntfamily \textcolor{red}{\textless type(varying\_string) \textgreater} type() Return the type of this
object.

ntfamily nodeComponentMergingStatistics

tfamily \void builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied
XML definition.

ntfamily \double zero density(<double(3)> positionSpherical→, <componentType> [compone-
typeType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→,
<logical> [haloLoaded]→) Compute the density.

ntfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

ntfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

ntfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

ntfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

ntfamily \double zero enclosedMass(<double> radius→, <componentType> [compone-
typeType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→,
<logical> [haloLoaded]→) Compute the mass enclosed within a radius.

ntfamily \textcolor{red}{\textless double \textgreater} galaxyMajorMergerTime() Get the galaxyMajorMergerTime
property of the mergingStatistics component.

ntfamily \textcolor{red}{\textless double\_type(varying\_string)\textgreater} galaxyMajorMergerTimeAttributeM-
atch(<logical> [requireGettable]→, <logical> [requireSettable]→, <logical> [requireEvolvable]→) Re-
turn a list of implementations that provide the given list off attributes for the galaxyMajorMergerTime
property of the mergingStatistics component

ntfamily \textcolor{red}{\textless double\_type} galaxyMajorMergerTimeIsGettable() Get the galaxyMajorMergerTime
property of the mergingStatistics component.

ntfamily \logical zero galaxyMajorMergerTimeIsSettable() Specify whether the galaxyMajorMergerTime
property of the mergingStatistics component is settable.
tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

tfamily \textcolor{red}{\textless double\textgreater} nodeFormationTime() Get the nodeFormationTime property of the mergingStatistics component.

tfamily \textcolor{red}{\textless type(varying\_string):\textgreater} nodeFormationTimeAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the nodeFormationTime property of the mergingStatistics component

tfamily \textcolor{red}{\textless integer\textgreater} nodeHierarchyLevel() Get the nodeHierarchyLevel property of the mergingStatistics component.

tfamily \textcolor{red}{\textless type(varying\_string):\textgreater} nodeHierarchyLevelAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the nodeHierarchyLevel property of the mergingStatistics component

tfamily \textcolor{red}{\textless integer\textgreater} nodeHierarchyLevelIsGettable() Get the nodeHierarchyLevel property of the mergingStatistics component.

tfamily \logicalzero nodeHierarchyLevelIsSettable() Specify whether the nodeHierarchyLevel property of the mergingStatistics component is settable.

tfamily \void nodeHierarchyLevelSet(<integer> value) Set the nodeHierarchyLevel property of the mergingStatistics component.

tfamily \textcolor{red}{\textless double\textgreater} nodeMajorMergerTime() Get the nodeMajorMergerTime property of the mergingStatistics component.

tfamily \textcolor{red}{\textless type(treeNode)\textgreater} hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void nodeFormationTimeSet(<double> value) Set the nodeFormationTime property of the mergingStatistics component.

tfamily \void nodeHierarchyLevelSet(<integer> value) Set the nodeHierarchyLevel property of the mergingStatistics component.

tfamily \void nodeMajorMergerTimeSet(<double> value) Set the nodeMajorMergerTime property of the mergingStatistics component.
tfamily \textcolor{red}{\textless\textgreater} \nodeMajorMergerTimeAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the nodeMajorMergerTime property of the mergingStatistics component.

tfamily \textcolor{red}{\textless\textgreater} \nodeMajorMergerTimeIsGettable() Get the nodeMajorMergerTime property of the mergingStatistics component.

tfamily \logicalzero \nodeMajorMergerTimeIsSettable() Specify whether the nodeMajorMergerTime property of the mergingStatistics component is settable.

tfamily \void \nodeMajorMergerTimeSet(<double> value) Set the nodeMajorMergerTime property of the mergingStatistics component.

tfamily \logicalzero \nullIsActive() Return whether the null implementation of the mergingStatistics component class is active.

tfamily \void \odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void \odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void \output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer> doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \void \outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \void \outputNames(<integer> integerProperty↔, <char[*](:)> integerPropertyNames↔, <char[*](:)> integerPropertyComments↔, <double(:)> integerPropertyUnitsSI↔, <integer> doubleProperty↔, <char[*](:)> doublePropertyNames↔, <char[*](:)> doublePropertyComments↔, <double(:)> doublePropertyUnitsSI↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \doublezero \potential(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the gravitational potential.

tfamily \logicalzero \recentIsActive() Return whether the recent implementation of the mergingStatistics component class is active.

tfamily \textcolor{red}{\textless\textgreater} \recentMajorMergerCount() Get the recentMajorMergerCount property of the mergingStatistics component.

tfamily \textcolor{red}{\textless\textgreater} \recentMajorMergerCountAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the recentMajorMergerCount property of the mergingStatistics component.

tfamily \textcolor{red}{\textless\textgreater} \recentMajorMergerCountIsGettable() Get the recentMajorMergerCount property of the mergingStatistics component.

tfamily \logicalzero \recentMajorMergerCountIsSettable() Specify whether the recentMajorMergerCount property of the mergingStatistics component is settable.

tfamily \void \recentMajorMergerCountSet(<integer(:)> value) Set the recentMajorMergerCount property of the mergingStatistics component.
tfamily \doublezero\ rotationCurve(<\ double > radius→, <\ componentType > [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

tfamily \doublezero\ rotationCurveGradient(<\ double > radius→, <\ componentType > [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the mergingStatistics component class is active.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical→, <\ componentType > [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<\ double > radius→, <\ componentType > [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless\ double\textgreater} galaxyMajorMergerTime() Get the galaxyMajorMergerTime property of the mergingStatistics component.

tfamily \textcolor{red}{\textless\double\textgreater\textless type(varying\_string):\textgreater} galaxyMajorMergerTimeAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the galaxyMajorMergerTime property of the mergingStatistics component.
tfamily {\textcolor{red}{\textless double\textgreater}} galaxyMajorMergerTimeIsGettable() Get the \texttt{galaxyMajorMergerTime} property of the \texttt{mergingStatistics} component.

tfamily \logicalzero galaxyMajorMergerTimeIsSettable() Specify whether the \texttt{galaxyMajorMergerTime} property of the \texttt{mergingStatistics} component is settable.

tfamily \void galaxyMajorMergerTimeSet(<double> value) Set the \texttt{galaxyMajorMergerTime} property of the \texttt{mergingStatistics} component.

tfamily {\textcolor{red}{\textless *type(treeNode)\textgreater}} host() Return a pointer to the host \texttt{treeNode} object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the \texttt{hotHaloCoolingAngularMomentum} property of the \texttt{hotHalo} component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the \texttt{hotHaloCoolingMass} property of the \texttt{hotHalo} component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count \rightarrow, <varying_string>name \leftarrow) Return the name of a property given is index.

tfamily {\textcolor{red}{\textless double\textgreater}} nodeFormationTime() Get the \texttt{nodeFormationTime} property of the \texttt{mergingStatistics} component.

tfamily {\textcolor{red}{\textless type(varying\_string)(:\textgreater)}} nodeFormationTimeAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the \texttt{nodeFormationTime} property of the \texttt{mergingStatistics} component.

tfamily {\textcolor{red}{\textless integer\textgreater}} nodeFormationTimeIsGettable() Get the \texttt{nodeFormationTime} property of the \texttt{mergingStatistics} component.

tfamily \logicalzero nodeFormationTimeIsSettable() Specify whether the \texttt{nodeFormationTime} property of the \texttt{mergingStatistics} component is settable.

tfamily \void nodeFormationTimeSet(<double> value) Set the \texttt{nodeFormationTime} property of the \texttt{mergingStatistics} component.

tfamily {\textcolor{red}{\textless integer\textgreater}} nodeHierarchyLevel() Get the \texttt{nodeHierarchyLevel} property of the \texttt{mergingStatistics} component.

tfamily {\textcolor{red}{\textless type(varying\_string)(:\textgreater)}} nodeHierarchyLevelAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the \texttt{nodeHierarchyLevel} property of the \texttt{mergingStatistics} component.

tfamily {\textcolor{red}{\textless integer\textgreater}} nodeHierarchyLevelIsGettable() Get the \texttt{nodeHierarchyLevel} property of the \texttt{mergingStatistics} component.

tfamily \logicalzero nodeHierarchyLevelIsSettable() Specify whether the \texttt{nodeHierarchyLevel} property of the \texttt{mergingStatistics} component is settable.
tfamily \void nodeHierarchyLevelSet(<integer> value) Set the nodeHierarchyLevel property of the mergingStatistics component.

tfamily \textcolor{red}{\textless\textgreater} nodeMajorMergerTime() Get the nodeMajorMergerTime property of the mergingStatistics component.

tfamily \textcolor{red}{\textless\textgreater} nodeMajorMergerTimeAttributeMatch(<logical> [requireGettable] →, <logical> [requireSettable] →, <logical> [requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the nodeMajorMergerTime property of the mergingStatistics component.

tfamily \textcolor{red}{\textless\textgreater} nodeMajorMergerTimeIsGettable() Get the nodeMajorMergerTime property of the mergingStatistics component.

tfamily \textcolor{red}{\textless\textgreater} nodeMajorMergerTimeIsSettable() Specify whether the nodeMajorMergerTime property of the mergingStatistics component is settable.

tfamily \void nodeMajorMergerTimeSet(<double> value) Set the nodeMajorMergerTime property of the mergingStatistics component.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the mergingStatistics component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer> integerBuffer ↔, <integer> doubleProperty ↔, <integer> doubleBufferCount ↔, <double> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty ↔, <char[]> integerPropertyNames ↔, <char[]> integerPropertyComments ↔, <double> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char[]> doublePropertyNames ↔, <char[]> doublePropertyComments ↔, <double> doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

tfamily \doublezero potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

tfamily \void readRaw(<integer> fileHandle ↔) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \logicalzero recentIsActive() Return whether the recent implementation of the mergingStatistics component class is active.

tfamily \textcolor{red}{\textless\textgreater} recentMajorMergerCount() Get the recentMajorMergerCount property of the mergingStatistics component.

tfamily \textcolor{red}{\textless\textgreater} recentMajorMergerCountAttributeMatch(<logical> [requireGettable] →, <logical> [requireSettable] →, <logical> [requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the recentMajorMergerCount property of the mergingStatistics component.
16.5. Objects

tfamily \textcolor{red}{\textless integer(:)\textgreater} recentMajorMergerCountIsGettable() Get the recentMajorMergerCount property of the mergingStatistics component.

ntfamily \textcolor{red}{\textless integer(:)\textgreater} recentMajorMergerCountIsSettable() Specify whether the recentMajorMergerCount property of the mergingStatistics component is settable.

ntfamily void recentMajorMergerCountSet(<integer(:)> value) Set the recentMajorMergerCount property of the mergingStatistics component.

ntfamily double zero rotationCurve(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the rotation curve.

ntfamily double zero rotationCurveGradient(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the rotation curve gradient.

ntfamily int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.

ntfamily void serializeRates(<double(:)> array \rightarrow) Serialize the evolvable rates to an array.

ntfamily void serializeScales(<double(:)> array \rightarrow) Serialize the evolvable scales to an array.

ntfamily void serializeValues(<double(:)> array \rightarrow) Serialize the evolvable quantities to an array.

ntfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

ntfamily nodeComponentMergingStatisticsRecent

ntfamily void builder(<type(node)>componentDefinition \rightarrow) Build a nodeComponent from a supplied XML definition.

ntfamily double zero density(<double(3)> positionSpherical \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <weightBy> [weightBy] \rightarrow, <integer> [weightIndex] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the density.

ntfamily void deserializeRates(<double(:)> array \leftarrow) Deserialize the evolvable rates from an array.

ntfamily void deserializeScales(<double(:)> array \leftarrow) Deserialize the evolvable scales from an array.

ntfamily void deserializeValues(<double(:)> array \leftarrow) Deserialize the evolvable quantities from an array.

ntfamily void destroy() Destroy the object.

ntfamily void dump() Generate an ASCII dump of all properties.

ntfamily void dumpRaw(<integer> fileHandle \rightarrow) Generate a binary dump of all properties.

ntfamily void dumpXML() Generate an XML dump of all properties.
Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless double\textgreater }galaxyMajorMergerTime() Get the \texttt{galaxyMajorMergerTime} property of the \texttt{mergingStatistics} component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)}galaxyMajorMergerTimeAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the \texttt{galaxyMajorMergerTime} property of the \texttt{mergingStatistics} component.

tfamily \textcolor{red}{\textless double\textgreater }galaxyMajorMergerTimeIsGettable() Get the \texttt{galaxyMajorMergerTime} property of the \texttt{mergingStatistics} component.

tfamily \logicalzero galaxyMajorMergerTimeIsSettable() Specify whether the \texttt{galaxyMajorMergerTime} property of the \texttt{mergingStatistics} component is settable.

tfamily void galaxyMajorMergerTimeSet(<double> value) Set the \texttt{galaxyMajorMergerTime} property of the \texttt{mergingStatistics} component.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater }host() Return a pointer to the host \texttt{treeNode} object.

tfamily void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component.

tfamily void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the \texttt{hotHaloCoolingAngularMomentum} property of the \texttt{hotHalo} component.

tfamily void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the \texttt{hotHaloCoolingMass} property of the \texttt{hotHalo} component.

tfamily void initialize() Initialize the object.

tfamily void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

tfamily \textcolor{red}{\textless double\textgreater }nodeFormationTime() Get the \texttt{nodeFormationTime} property of the \texttt{mergingStatistics} component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)}nodeFormationTimeAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the \texttt{nodeFormationTime} property of the \texttt{mergingStatistics} component.

tfamily \textcolor{red}{\textless double\textgreater }nodeFormationTimeIsGettable() Get the \texttt{nodeFormationTime} property of the \texttt{mergingStatistics} component.

tfamily \logicalzero nodeFormationTimeIsSettable() Specify whether the \texttt{nodeFormationTime} property of the \texttt{mergingStatistics} component is settable.

tfamily void nodeFormationTimeSet(<double> value) Set the \texttt{nodeFormationTime} property of the \texttt{mergingStatistics} component.
16.5. Objects

tfamily \textcolor{red}{\textless integer\textgreater} nodeHierarchyLevel() Get the nodeHierarchyLevel property of the mergingStatistics component.

tfamily \textcolor{red}{\textless integer\textgreater} nodeHierarchyLevelAttributeMatch(\textless logical\[requireGettable\]→, \textless logical\[requireSettable\]→, \textless logical\[requireEvolvable\]→) Return a list of implementations that provide the given list of attributes for the nodeHierarchyLevel property of the mergingStatistics component.

tfamily \textcolor{red}{\textless integer\textgreater} nodeHierarchyLevelIsGettable() Get the nodeHierarchyLevel property of the mergingStatistics component.

tfamily \textcolor{red}{\textless integer\textgreater} nodeHierarchyLevelIsSettable() Specify whether the nodeHierarchyLevel property of the mergingStatistics component is settable.

tfamily \textcolor{red}{\textless integer\textgreater} nodeHierarchyLevelSet(<integer> value) Set the nodeHierarchyLevel property of the mergingStatistics component.

tfamily \textcolor{red}{\textless double\textgreater} nodeMajorMergerTime() Get the nodeMajorMergerTime property of the mergingStatistics component.

tfamily \textcolor{red}{\textless double\textgreater} nodeMajorMergerTimeAttributeMatch(\textless logical\[requireGettable\]→, \textless logical\[requireSettable\]→, \textless logical\[requireEvolvable\]→) Return a list of implementations that provide the given list of attributes for the nodeMajorMergerTime property of the mergingStatistics component.

tfamily \textcolor{red}{\textless double\textgreater} nodeMajorMergerTimeIsGettable() Get the nodeMajorMergerTime property of the mergingStatistics component.

tfamily \textcolor{red}{\textless double\textgreater} nodeMajorMergerTimeIsSettable() Specify whether the nodeMajorMergerTime property of the mergingStatistics component is settable.

tfamily \textcolor{red}{\textless double\textgreater} nodeMajorMergerTimeSet(<double> value) Set the nodeMajorMergerTime property of the mergingStatistics component.

tfamily \textcolor{red}{\textless logical\textgreater} nullIsActive() Return whether the null implementation of the mergingStatistics component class is active.

tfamily \textcolor{red}{\textless logical\textgreater} odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \textcolor{red}{\textless logical\textgreater} odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \textcolor{red}{\textless integer\textgreater \textless integer\textgreater \textless integer\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless integer\textgreater \textless integer\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\textgreater \textless double\text greater

\[595\]
function readRaw(fileHandle) 
    Read a binary dump of the nodeComponent from the given fileHandle.

function isActive() 
    Return whether the recent implementation of the mergingStatistics component class is active.

function recentMajorMergerCount() 
    Get the recentMajorMergerCount property of the mergingStatistics component.

function recentMajorMergerCountAttributeMatch() 
    Return a list of implementations that provide the given list of attributes for the recentMajorMergerCount property of the mergingStatistics component.

function recentMajorMergerCountIsGettable() 
    Get the recentMajorMergerCount property of the mergingStatistics component.

function recentMajorMergerCountIsSettable() 
    Specify whether the recentMajorMergerCount property of the mergingStatistics component is settable.

function recentMajorMergerCountSet(value) 
    Set the recentMajorMergerCount property of the mergingStatistics component.

function rotationCurve(radius, componentType, massType, haloLoaded) 
    Compute the rotation curve.

function rotationCurveGradient(radius, componentType, massType, haloLoaded) 
    Compute the rotation curve gradient.

function serializeCount() 
    Return a count of the number of evolvable quantities to be evolved.

function serializeRates(array) 
    Serialize the evolvable rates to an array.

function serializeScales(array) 
    Serialize the evolvable scales to an array.

function serializeValues(array) 
    Serialize the evolvable quantities to an array.

function standardIsActive() 
    Return whether the standard implementation of the mergingStatistics component class is active.

function surfaceDensity(positionCylindrical, componentType, massType, haloLoaded) 
    Compute the surface density.

function type() 
    Return the type of this object.

function nodeComponentMergingStatisticsStandard 
    Build a nodeComponent from a supplied XML definition.
16.5. Objects

tfamily \void deserializeRates(<double:>) array← Deserialize the evolvable rates from an array.
tfamily \void deserializeScales(<double:>) array← Deserialize the evolvable scales from an array.
tfamily \void deserializeValues(<double:>) array← Deserialize the evolvable quantities from an array.
tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.
tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.
tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \double zeroEnclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless double\textgreater} galaxyMajorMergerTime() Get the galaxyMajorMergerTime property of the mergingStatistics component.

tfamily \textcolor{red}{\textless double\textgreater} galaxyMajorMergerTimeAttributeMatch(<logical> [requireGettable]→, <logical> [requireSettable]→, <logical> [requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the galaxyMajorMergerTime property of the mergingStatistics component.

tfamily \textcolor{red}{\textless double\textgreater} galaxyMajorMergerTimeIsGettable() Get the galaxyMajorMergerTime property of the mergingStatistics component.

tfamily \logical zero galaxyMajorMergerTimeIsSettable() Specify whether the galaxyMajorMergerTime property of the mergingStatistics component is settable.

tfamily \void galaxyMajorMergerTimeSet(<double> value) Set the galaxyMajorMergerTime property of the mergingStatistics component.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count→, <varying_string> name←) Return the name of a property given is index.

tfamily \textcolor{red}{\textless double\textgreater} nodeFormationTime() Get the nodeFormationTime property of the mergingStatistics component.
tfamily \textcolor{red}{\textless double \textgreater} nodeFormationTime() Get the \textit{nodeFormationTime} property of the \textit{mergingStatistics} component.

tfamily \textcolor{red}{\textless integer \textgreater} nodeHierarchyLevel() Get the \textit{nodeHierarchyLevel} property of the \textit{mergingStatistics} component.

tfamily \textcolor{red}{\textless double \textgreater} nodeMajorMergerTime() Get the \textit{nodeMajorMergerTime} property of the \textit{mergingStatistics} component.

598
16.5. Objects

tfamily void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

tfamily void outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily void outputNames(<integer> integerProperty ↔, <char[*](:) integerPropertyNames ↔, <char[*](:) integerPropertyComments ↔, <double(:)> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char[*](:) doublePropertyNames ↔, <char[*](:) doublePropertyComments ↔, <double(:)> doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

tfamily double zero potential(<double> radius →, <componentType>[componentType] →, <massType>[massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

tfamily void readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily logical zero recentIsActive() Return whether the recent implementation of the mergingStatistics component class is active.

tfamily textcolor{red}{textless integer(:)}textgreater recentMajorMergerCount() Get the recentMajorMergerCount property of the mergingStatistics component.

tfamily textcolor{red}{textless type(varying \_string)(:)textgreater} recentMajorMergerCountAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the recentMajorMergerCount property of the mergingStatistics component.

tfamily textcolor{red}{textless integer(:)}textgreater recentMajorMergerCountIsGettable() Get the recentMajorMergerCount property of the mergingStatistics component.

tfamily logical zero recentMajorMergerCountIsSettable() Specify whether the recentMajorMergerCount property of the mergingStatistics component is settable.

tfamily void recentMajorMergerCountSet(<integer(:)> value) Set the recentMajorMergerCount property of the mergingStatistics component.

tfamily double zero rotationCurve(<double> radius →, <componentType>[componentType] →, <massType>[massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

tfamily double zero rotationCurveGradient(<double> radius →, <componentType>[componentType] →, <massType>[massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

tfamily int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily void serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

tfamily void serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

tfamily void serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

tfamily logical zero standardIsActive() Return whether the standard implementation of the mergingStatistics component class is active.
16. Coding GALACTICUS

tfamily double zero surfaceDensity(<double(3)> positionCylindrical\rightarrow, <componentType> \rightarrow, <massType> \rightarrow, <weightBy> \rightarrow, <integer> \rightarrow, <logical> \rightarrow) Compute the surface density.

tfamily text color{red} {textless type(varying\_string)\textgreater} type() Return the type of this object.

ntfamily nodeComponentNBody

tfamily void builder(\*type(node)>componentDefinition\rightarrow) Build a nodeComponent from a supplied XML definition.

tfamily double zero density(<double(3)> positionSpherical\rightarrow, <componentType> \rightarrow, <massType> \rightarrow, <weightBy> \rightarrow, <integer> \rightarrow, <logical> \rightarrow) Compute the density.

tfamily void deserializeRates(<double(:)> array\leftarrow) Deserialize the evolvable rates from an array.

tfamily void deserializeScales(<double(:)> array\leftarrow) Deserialize the evolvable scales from an array.

tfamily void deserializeValues(<double(:)> array\leftarrow) Deserialize the evolvable quantities from an array.

tfamily void destroy() Destroy the object.

tfamily void dump() Generate an ASCII dump of all properties.

tfamily void dumpRaw(<integer> fileHandle\rightarrow) Generate a binary dump of all properties.

tfamily void dumpXML() Generate an XML dump of all properties.

tfamily double zero enclosedMass(<double> radius\rightarrow, <componentType> \rightarrow, <massType> \rightarrow, <weightBy> \rightarrow, <integer> \rightarrow, <logical> \rightarrow) Compute the mass enclosed within a radius.

tfamily text color{red} {textless \*type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily void hotHalo CoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHalo CoolingAbundances property of the hotHalo component.

tfamily void hotHalo CoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHalo CoolingAngularMomentum property of the hotHalo component.

tfamily void hotHalo CoolingMassRate(<double> value) Cumulate to the rate of the hotHalo CoolingMass property of the hotHalo component.

tfamily void initialize() Initialize the object.

tfamily void nameFromIndex(<integer> count\rightarrow, <varying_string> name\leftarrow) Return the name of a property given is index.

tfamily logical zero nullIsActive() Return whether the null implementation of the nBody component class is active.

tfamily void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily void odeStepScalesInitialize() Initialize scales for evolvable properties.
16.5. Objects

tfamily \void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer> doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty ↔, <char[*](:)> integerPropertyNames ↔, <char[*](:)> integerPropertyComments ↔, <double(:)> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char[*](:)> doublePropertyNames ↔, <char[*](:)> doublePropertyComments ↔, <double(:)> doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

tfamily \textcolor{red}{\textless integer(kind=kind_int8)\textgreater} particleCount() Get the particleCount property of the nBody component.

tfamily \textcolor{red}{\textless type(varying_string)(:}\textgreater\textcolor{red}{\textgreater}} particleCountAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the particleCount property of the nBody component.

tfamily \textcolor{red}{\textless integer(kind=kind_int8)\textgreater} particleCountIsGettable() Get the particleCount property of the nBody component.

tfamily \logical zero particleCountIsSettable() Specify whether the particleCount property of the nBody component is settable.

tfamily \void particleCountSet(<integer(kind=kind_int8)> value) Set the particleCount property of the nBody component.

tfamily \double zero potential(<double> radius →, <componentType>[componentType] →, <massType>[massType] →, <logical>[haloLoaded] →) Compute the gravitational potential.

tfamily \double zero rotationCurve(<double> radius →, <componentType>[componentType] →, <massType>[massType] →, <logical>[haloLoaded] →) Compute the rotation curve.

tfamily \double zero rotationCurveGradient(<double> radius →, <componentType>[componentType] →, <massType>[massType] →, <logical>[haloLoaded] →) Compute the rotation curve gradient.

tfamily \int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

tfamily \logical zero standardIsActive() Return whether the standard implementation of the nBody component class is active.

tfamily \double zero surfaceDensity(<double(3)> positionCylindrical →, <componentType>[componentType] →, <massType>[massType] →, <weightBy>[weightBy] →, <integer>[weightIndex] →, <logical>[haloLoaded] →) Compute the surface density.

tfamily \textcolor{red}{\textless type(varying_string)\textgreater} type() Return the type of this object.
tfamily \textcolor{red}{\textless double\textgreater} velocityDispersion() Get the \texttt{velocityDispersion} property of the \texttt{nBody} component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater velocityDispersionAttributeMatch(<\texttt{logical}>\texttt{requireGettable}→, <\texttt{logical}>\texttt{requireSettable}→, <\texttt{logical}>\texttt{requireEvolvable}→) Return a list of implementations that provide the given list of attributes for the \texttt{velocityDispersion} property of the \texttt{nBody} component.

tfamily \textcolor{red}{\textless double\textgreater} velocityDispersionIsGettable() Get the \texttt{velocityDispersion} property of the \texttt{nBody} component.

tfamily \texttt{void velocityDispersionSet(<double> value)} Set the \texttt{velocityDispersion} property of the \texttt{nBody} component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater velocityMaximumAttributeMatch(<\texttt{logical}>\texttt{requireGettable}→, <\texttt{logical}>\texttt{requireSettable}→, <\texttt{logical}>\texttt{requireEvolvable}→) Return a list of implementations that provide the given list of attributes for the \texttt{velocityMaximum} property of the \texttt{nBody} component.

tfamily \textcolor{red}{\textless double\textgreater} velocityMaximum() Get the \texttt{velocityMaximum} property of the \texttt{nBody} component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)}\textgreater velocityMaximumIsGettable() Get the \texttt{velocityMaximum} property of the \texttt{nBody} component.

tfamily \texttt{void velocityMaximumSet(<double> value)} Set the \texttt{velocityMaximum} property of the \texttt{nBody} component.

tfamily \texttt{nodeComponentNBodyNull}

tfamily \texttt{void builder(<\texttt{type(node)}\textgreater componentDefinition→)} Build a \texttt{nodeComponent} from a supplied XML definition.

tfamily \texttt{double density(<double(3)\textgreater positionSpherical→, <\texttt{componentType} [\texttt{componentType}]→, <\texttt{massType} [\texttt{massType}]→, <\texttt{weightBy} [\texttt{weightBy}]→, <\texttt{integer} [\texttt{weightIndex}]→, <\texttt{logical} [\texttt{haloLoaded}]→) Compute the density.

tfamily \texttt{void deserializeRates(<double(:) array←)} Deserialize the evolvable rates from an array.

tfamily \texttt{void deserializeScales(<double(:) array←)} Deserialize the evolvable scales from an array.

tfamily \texttt{void deserializeValues(<double(:) array←)} Deserialize the evolvable quantities from an array.

tfamily \texttt{void destroy()} Destroy the object.

tfamily \texttt{void dump()} Generate an ASCII dump of all properties.

tfamily \texttt{void dumpRaw(<integer> fileHandle→)} Generate a binary dump of all properties.

tfamily \texttt{void dumpXML()} Generate an XML dump of all properties.
16.5. Objects

tfamily\doublezeroenclosedMass(\text{\textless} double\textgreater{} radius\rightarrow, \text{\textless} componentType\textgreater{} [\text{\textless} componentType\textgreater{}]\rightarrow, \text{\textless} massType\textgreater{} [\text{\textless} massType\textgreater{}]\rightarrow, \text{\textless} weightBy\textgreater{} [\text{\textless} weightBy\textgreater{}]\rightarrow, \text{\textless} integer\textgreater{} [\text{\textless} weightIndex\textgreater{}]\rightarrow, \text{\textless} logical\textgreater{} [\text{\textless} haloLoaded\textgreater{}]\rightarrow)
Compute the mass enclosed within a radius.

tfamily\textcolor{red}{\textless}type(treeNode)\textgreater{} host() Return a pointer to the host treeNode object.

tfamily\void hotHaloCoolingAbundancesRate(\text{\textless} type(abundances)\textgreater{} value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily\void hotHaloCoolingAngularMomentumRate(\text{\textless} double\textgreater{} value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily\void hotHaloCoolingMassRate(\text{\textless} double\textgreater{} value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily\void initialize() Initialize the object.

tfamily\void nameFromIndex(\text{\textless} integer\textgreater{} count\rightarrow, \text{\textless} varying_string\textgreater{} name\leftarrow) Return the name of a property given its index.

tfamily\logicalzero nullIsActive() Return whether the null implementation of the nBody component class is active.

tfamily\void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily\void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily\void output(\text{\textless} integer\textgreater{} integerProperty\leftrightarrow, \text{\textless} integer\textgreater{} integerBufferCount\leftrightarrow, \text{\textless} integer()\textgreater{} integerBuffer\leftrightarrow, \text{\textless} integer\textgreater{} doubleProperty\leftrightarrow, \text{\textless} integer\textgreater{} doubleBufferCount\leftrightarrow, \text{\textless} double()\textgreater{} doubleBuffer\leftrightarrow, \text{\textless} double\textgreater{} time\rightarrow, \text{\textless} integer\textgreater{} instance\rightarrow) Generate values of outputtable properties.

tfamily\void outputCount(\text{\textless} integer\textgreater{} integerPropertyCount\leftrightarrow, \text{\textless} integer\textgreater{} doublePropertyCount\leftrightarrow, \text{\textless} double\textgreater{} time\rightarrow, \text{\textless} integer\textgreater{} instance\rightarrow) Compute a count of outputtable properties.

tfamily\void outputNames(\text{\textless} integer\textgreater{} integerProperty\leftrightarrow, \text{\textless} char[*]()\textgreater{} integerPropertyNames\leftrightarrow, \text{\textless} char[*]()\textgreater{} integerPropertyComments\leftrightarrow, \text{\textless} double()\textgreater{} integerPropertyUnitsSI\leftrightarrow, \text{\textless} integer\textgreater{} doubleProperty\leftrightarrow, \text{\textless} char[*]()\textgreater{} doublePropertyNames\leftrightarrow, \text{\textless} char[*]()\textgreater{} doublePropertyComments\leftrightarrow, \text{\textless} double()\textgreater{} doublePropertyUnitsSI\leftrightarrow, \text{\textless} double\textgreater{} time\rightarrow, \text{\textless} integer\textgreater{} instance\rightarrow) Generate names of outputtable properties.

tfamily\textcolor{red}{\textless}integer(kind=kind_int8)\textgreater{} particleCount() Get the particleCount property of the nBody component.

tfamily\textcolor{red}{\textless}type(varying\_\textless varying_string\textgreater{}(\textgreater{})\textgreater{} particleCountAttributeMatch(\text{\textless} logical\textgreater{}[\text{\textless} requireGettable\textgreater{}\rightarrow, \text{\textless} logical\textgreater{}[\text{\textless} requireSettable\textgreater{}\rightarrow, \text{\textless} logical\textgreater{}[\text{\textless} requireEvolvable\textgreater{}\rightarrow]) Return a list of implementations that provide the given list of attributes for the particleCount property of the nBody component.

tfamily\textcolor{red}{\textless}integer(kind=kind_int8)\textgreater{} particleCountIsGettable() Get the particleCount property of the nBody component.

tfamily\logicalzero particleCountIsSettable() Specify whether the particleCount property of the nBody component is settable.

tfamily\void particleCountSet(\text{\textless} integer(kind=kind_int8)\textgreater{} value) Set the particleCount property of the nBody component.
16. Coding GALACTICUS

tfamily `\doublezero potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→)` Compute the gravitational potential.

tfamily `\void readRaw(<integer> fileHandle→)` Read a binary dump of the `nodeComponent` from the given `fileHandle`.

tfamily `\doublezero rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→)` Compute the rotation curve.

tfamily `\doublezero rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→)` Compute the rotation curve gradient.

tfamily `\intzero serializeCount()` Return a count of the number of evolvable quantities to be evolved.

tfamily `\void serializeRates(<double(:)> array→)` Serialize the evolvable rates to an array.

tfamily `\void serializeScales(<double(:)> array→)` Serialize the evolvable scales to an array.

tfamily `\void serializeValues(<double(:)> array→)` Serialize the evolvable quantities to an array.

tfamily `\logicalzero standardIsActive()` Return whether the standard implementation of the `nBody` component class is active.

tfamily `\doublezero surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)` Compute the surface density.

tfamily `\textcolor{red}{\textless type(varying\_string)\textgreater} type()` Return the type of this object.

tfamily `\textcolor{red}{\textless double\textgreater} velocityDispersion()` Get the `velocityDispersion` property of the `nBody` component.

tfamily `\textcolor{red}{\textless type(varying\_string)(:)\textgreater} velocityDispersionAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→)` Return a list of implementations that provide the given list of attributes for the `velocityDispersion` property of the `nBody` component.

tfamily `\textcolor{red}{\textless double\textgreater} velocityDispersionIsGettable()` Get the `velocityDispersion` property of the `nBody` component.

tfamily `\logicalzero velocityDispersionIsSettable()` Specify whether the `velocityDispersion` property of the `nBody` component is settable.

tfamily `\void velocityDispersionSet(<double> value)` Set the `velocityDispersion` property of the `nBody` component.

tfamily `\textcolor{red}{\textless double\textgreater} velocityMaximum()` Get the `velocityMaximum` property of the `nBody` component.

tfamily `\textcolor{red}{\textless type(varying\_string)(:)\textgreater} velocityMaximumAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→)` Return a list of implementations that provide the given list of attributes for the `velocityMaximum` property of the `nBody` component.
tfamily \textcolor{red}{\{\textless\texttt{double}\textgreater\} velocityMaximumIsGettable() Get the velocityMaximum property of the \texttt{nBody} component.

tfamily \textcolor{red}{\textless\texttt{logical}\textgreater\} velocityMaximumIsSettable() Specify whether the velocityMaximum property of the \texttt{nBody} component is settable.

tfamily void velocityMaximumSet(<double> value) Set the velocityMaximum property of the \texttt{nBody} component.

tfamily nodeComponentNBodyStandard

tfamily \textcolor{red}{\textless\texttt{type(node)}\textgreater host() Return a pointer to the host \texttt{treeNode} object.

tfamily \textcolor{red}{\textless\texttt{logical}\textgreater\} nullIsActive() Return whether the null implementation of the \texttt{nBody} component class is active.

tfamily void odeStepRatesInitialize() Initialize rates for evolvable properties.
16. Coding GALACTICUS

tfamily | void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily | void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily | void outputCount(<integer>integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily | void outputNames(<integer>integerProperty↔, <char[*](::)>integerPropertyNames↔, <char[*](::)>integerPropertyComments↔, <double(:)>integerPropertyUnitsSI↔, <integer>doubleProperty ↔, <char[*](::)> doublePropertyNames ↔, <char[*](::)> doublePropertyComments ↔, <double(:)> doublePropertyUnitsSI ↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily | textcolor {red} {\textless integer(kind=kind\_int8)\textgreater} particleCount() Get the particleCount property of the nBody component.

tfamily | textcolor {red} {\textless varchar\_field(varying\_string)(::)\textgreater} particleCountAttributeMatch(<logical> [requireGettable]→, <logical> [requireSettable]→, <logical> [requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the particleCount property of the nBody component.

tfamily | textcolor {red} {\textless integer(kind=kind\_int8)\textgreater} particleCountIsGettable() Get the particleCount property of the nBody component.

tfamily | logical zero particleCountIsSettable() Specify whether the particleCount property of the nBody component is settable.

tfamily | void particleCountSet(<integer(kind=kind\_int8)> value) Set the particleCount property of the nBody component.

tfamily | double zero potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

tfamily | void readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily | double zero rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

tfamily | double zero rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

tfamily | int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily | void serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

tfamily | void serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

tfamily | void serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

tfamily | logical zero standardIsActive() Return whether the standard implementation of the nBody component class is active.

tfamily | double zero surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.
16.5. Objects

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

tfamily \textcolor{red}{\textless double\textgreater} velocityDispersion() Get the velocityDispersion property of the nBody component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} velocityDispersionAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the velocityDispersion property of the nBody component.

tfamily \textcolor{red}{\textless double\textgreater} velocityDispersionIsGettable() Get the velocityDispersion property of the nBody component.

Specify whether the velocityDispersion property of the nBody component is settable.

tfamily \void velocityDispersionSet(<double> value) Set the velocityDispersion property of the nBody component.

Compute the density.

tfamily \void deserializesRates(<double(:)> array←) Deserialize the evolvable rates from an array.

Deserializes the evolvable scales from an array.

tfamily \void deserializesValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.
16. Coding GALACTICUS

tfamily \void dumpRaw(<integer> fileHandle) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \double zero enclosedMass(<double> radius, <componentType> [componentType], <massType> [massType], <weightBy> [weightBy], <integer> [weightIndex], <logical> [haloLoaded]) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless type(treeNode)\textgreater} host() Return a pointer to the host \texttt{treeNode} object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the \texttt{hotHaloCoolingAngularMomentum} property of the \texttt{hotHalo} component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the \texttt{hotHaloCoolingMass} property of the \texttt{hotHalo} component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count, <varying_string> name) Return the name of a property given is index.

tfamily \logical zero nullIsActive() Return whether the null implementation of the position component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty, <integer> integerBufferCount, <integer(:,:)> integerBuffer, <integer> doubleProperty, <integer(:,:)> doubleBuffer, <double> time, <integer> instance) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount, <integer> doublePropertyCount, <double> time, <integer> instance) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty, <char[*](:)> integerPropertyNames, <char[*](:)> integerPropertyComments, <double(:)> integerPropertyUnitsSI, <integer> doubleProperty, <char[*](:)> doublePropertyNames, <char[*](:)> doublePropertyComments, <double(:)> doublePropertyUnitsSI, <double> time, <integer> instance) Generate names of outputtable properties.


tfamily \textcolor{red}{\textless double(:\textgreater)} position() Get the \texttt{position} property of the \texttt{position} component.

tfamily \textcolor{red}{\textless type(varying_string)\textgreater} positionAttributeMatch(<logical>[requireGettable], <logical>[requireSettable], <logical>[requireEvolvable]) Return a list of implementations that provide the given list off attributes for the \texttt{position} property of the \texttt{position} component.


tfamily \textcolor{red}{\textless history\textgreater} positionHistory() Get the \texttt{positionHistory} property of the \texttt{position} component.
16.5. Objects

Return a list of implementations that provide the given list of attributes for the \texttt{positionHistory} property of the \texttt{position} component.

Get the \texttt{positionHistory} property of the \texttt{position} component.

Specify whether the \texttt{positionHistory} property of the \texttt{position} component is settable.

Set the \texttt{positionHistory} property of the \texttt{position} component.

Specify whether the \texttt{positionHistory} property of the \texttt{position} component is settable.

Set the \texttt{position} property of the \texttt{position} component.

Specify whether the \texttt{position} property of the \texttt{position} component is settable.

Set the \texttt{position} property of the \texttt{position} component.

Return whether the preset implementation of the \texttt{position} component class is active.

Compute the gravitational potential.

Compute the rotation curve.

Compute the rotation curve gradient.

Return a count of the number of evolvable quantities to be evolved.

Serialize the evolvable rates to an array.

Serialize the evolvable scales to an array.

Serialize the evolvable quantities to an array.

Compute the surface density.

Return the type of this object.

Get the \texttt{velocity} property of the \texttt{position} component.

Return a list of implementations that provide the given list of attributes for the \texttt{velocity} property of the \texttt{position} component.

Get the \texttt{velocity} property of the \texttt{position} component.
tfamily \logicalzero velocityIsSettable() Specify whether the velocity property of the position component is settable.

tfamily \void velocitySet(<double()> value) Set the velocity property of the position component.

**tfamily nodeComponentPositionNull**

**tfamily \void builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.**

**tfamily \doublezero density(<double(3)> positionSpherical→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the density.**

**tfamily \void deserializeRates(<double()> array←) Deserialize the evolvable rates from an array.**

**tfamily \void deserializeScales(<double()> array←) Deserialize the evolvable scales from an array.**

**tfamily \void deserializeValues(<double()> array←) Deserialize the evolvable quantities from an array.**

**tfamily \void destroy() Destroy the object.**

**tfamily \void dump() Generate an ASCII dump of all properties.**

**tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.**

**tfamily \void dumpXML() Generate an XML dump of all properties.**

**tfamily \doublezero enclosedMass(<double>radius→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the mass enclosed within a radius.**

**tfamily \textcolor{red}{{textless}*type(treeNode){textgreater} host() Return a pointer to the host treeNode object.**

**tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.**

**tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.**

**tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.**

**tfamily \void initialize() Initialize the object.**

**tfamily \void nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.**

**tfamily \logicalzero nullIsActive() Return whether the null implementation of the position component class is active.**

**tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.**

**tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.**
16.5. Objects

tfamily \void output(<\text{integer}> integerProperty↔, <\text{integer}> integerBufferCount↔, <\text{integer}(::)> integerBuffer↔, <\text{integer}> doubleProperty↔, <\text{integer}> doubleBufferCount↔, <\text{double}(::)> doubleBuffer↔, <\text{double}> time→, <\text{integer}> instance→) Generate values of outputtable properties.

tfamily \void outputCount(<\text{integer}> integerPropertyCount↔, <\text{integer}> doublePropertyCount↔, <\text{double}> time→, <\text{integer}> instance→) Compute a count of outputtable properties.

tfamily \void outputNames(<\text{integer}> integerProperty↔, <\text{char}\[*](::)> integerPropertyNames↔, <\text{char}\[*](::)> integerPropertyComments↔, <\text{double}(::)> integerPropertyUnitsSI↔, <\text{integer}> doubleProperty↔, <\text{char}\[*](::)> doublePropertyNames↔, <\text{char}\[*](::)> doublePropertyComments↔, <\text{double}(::)> doublePropertyUnitsSI↔, <\text{double}> time→, <\text{integer}> instance→) Generate names of outputtable properties.

tfamily \textcolor{red}{\textless \text{double}(::)\textgreater} position() Get the position property of the position component.

tfamily \textcolor{red}{\textless \text{type}(\text{varying_string})(::)\textgreater} positionAttributeMatch(<\text{logical}>\[requireGettable\]→, <\text{logical}>\[requireSettable\]→, <\text{logical}>\[requireEvolvable\]→) Return a list of implementations that provide the given list off attributes for the position property of the position component.

tfamily \textcolor{red}{\textless \text{type}(\text{history})(::)\textgreater} positionHistory() Get the positionHistory property of the position component.

tfamily \textcolor{red}{\textless \text{type}(\text{history})(::)\textgreater} positionHistoryAttributeMatch(<\text{logical}>\[requireGettable\]→, <\text{logical}>\[requireSettable\]→, <\text{logical}>\[requireEvolvable\]→) Return a list of implementations that provide the given list off attributes for the positionHistory property of the position component.

tfamily \textcolor{red}{\textless \text{type}(\text{history})(::)\textgreater} positionHistoryIsGettable() Get the positionHistory property of the position component.

tfamily \textcolor{red}{\textless \text{type}(\text{history})(::)\textgreater} positionHistoryIsSettable() Specify whether the positionHistory property of the position component is settable.

tfamily \void positionHistorySet(<\text{type}(\text{history})> value) Set the positionHistory property of the position component.

tfamily \textcolor{red}{\textless \text{double}(::)\textgreater} positionIsGettable() Get the position property of the position component.

tfamily \textcolor{red}{\textless \text{double}(::)\textgreater} positionIsSettable() Specify whether the position property of the position component is settable.

tfamily \void positionSet(<\text{double}(::)> value) Set the position property of the position component.

tfamily \doublezero potential(<\text{double}> radius→, <\text{componentType}>\[componentType\]→, <\text{massType}>\[massType\]→, <\text{logical}>\[haloLoaded\]→) Compute the gravitational potential.

tfamily \textcolor{red}{\textless \text{logical}\textgreater} presetIsActive() Return whether the preset implementation of the position component class is active.

tfamily \void readRaw(<\text{integer}> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezero rotationCurve(<\text{double}> radius→, <\text{componentType}>\[componentType\]→, <\text{massType}>\[massType\]→, <\text{logical}>\[haloLoaded\]→) Compute the rotation curve.
Galacticus

### Coding

#### Functions

- **rotationCurveGradient**: Computes the rotation curve gradient.
  ```
  tfamily \textcolor{red}{\textless double\textgreater\ rotationCurveGradient( <double> radius\rightarrow , <componentType> \rightarrow [componentType]\rightarrow , <massType> \rightarrow [massType]\rightarrow , <logical> \rightarrow [haloLoaded]\rightarrow )}
  ```

- **serializeCount**: Returns a count of the number of evolvable quantities to be evolved.
  ```
  tfamily \textcolor{red}{\textless double\textgreater\ serializeCount()}
  ```

- **serializeRates**: Serializes the evolvable rates to an array.
  ```
  tfamily \textcolor{red}{\textless double(:)\ serializeRates( <double(:)> array\rightarrow )}
  ```

- **serializeScales**: Serializes the evolvable scales to an array.
  ```
  tfamily \textcolor{red}{\textless double(:)\ serializeScales( <double(:)> array\rightarrow )}
  ```

- **serializeValues**: Serializes the evolvable quantities to an array.
  ```
  tfamily \textcolor{red}{\textless double(:)\ serializeValues( <double(:)> array\rightarrow )}
  ```

- **surfaceDensity**: Computes the surface density.
  ```
  tfamily \textcolor{red}{\textless double(3)>\ surfaceDensity( <double(3)> positionCylindrical\rightarrow , <componentType> \rightarrow , <massType> \rightarrow , <weightBy> \rightarrow , <integer> \rightarrow , <logical> \rightarrow [haloLoaded]\rightarrow )}
  ```

- **type**: Returns the type of this object.
  ```
  tfamily \textcolor{red}{\textless type(varying_string)>\ type()}
  ```

- **velocity**: Gets the velocity property of the position component.
  ```
  tfamily \textcolor{red}{\textless double(:)>\ velocity()}
  ```

- **velocityAttributeMatch**: Returns a list of implementations that provide the given list off attributes for the velocity property of the position component.
  ```
  tfamily \textcolor{red}{\textless type(varying_string)(:)>\ velocityAttributeMatch( <logical>[requireGettable]\rightarrow , <logical>[requireSettable]\rightarrow , <logical>[requireEvolvable]\rightarrow )}
  ```

- **velocityIsGettable**: Gets the velocity property of the position component.
  ```
  tfamily \textcolor{red}{\textless double(:)>\ velocityIsGettable()}
  ```

- **velocityIsSettable**: Specifies whether the velocity property of the position component is settable.
  ```
  tfamily \textcolor{red}{\textless logical\textgreater\ velocityIsSettable()}
  ```

- **velocitySet**: Sets the velocity property of the position component.
  ```
  tfamily \textcolor{red}{\textless double(:)>\ velocitySet( <double(:)> value\rightarrow )}
  ```

- **nodeComponentPositionPreset**: Builds a nodeComponent from a supplied XML definition.
  ```
  tfamily \textcolor{red}{\textless *type(node)>\ nodeComponentPositionPreset( <type(node)> componentDefinition\rightarrow )}
  ```

- **density**: Computes the density.
  ```
  tfamily \textcolor{red}{\textless double(3)>\ density( <double(3)> positionSpherical\rightarrow , <componentType> \rightarrow , <massType> \rightarrow , <weightBy> \rightarrow , <integer> \rightarrow , <logical> \rightarrow [haloLoaded]\rightarrow )}
  ```

- **deserializeRates**: Deserializes the evolvable rates from an array.
  ```
  tfamily \textcolor{red}{\textless double(:)>\ deserializeRates( <double(:)> array\rightarrow )}
  ```

- **deserializeScales**: Deserializes the evolvable scales from an array.
  ```
  tfamily \textcolor{red}{\textless double(:)>\ deserializeScales( <double(:)> array\rightarrow )}
  ```

- **deserializeValues**: Deserializes the evolvable quantities from an array.
  ```
  tfamily \textcolor{red}{\textless double(:)>\ deserializeValues( <double(:)> array\rightarrow )}
  ```

- **destroy**: Destroys the object.
  ```
  tfamily \textcolor{red}{\textless logical\textgreater\ destroy()}
  ```

- **dump**: Generates an ASCII dump of all properties.
  ```
  tfamily \textcolor{red}{\textless logical\textgreater\ dump()}
  ```

- **dumpRaw**: Generates a binary dump of all properties.
  ```
  tfamily \textcolor{red}{\textless integer\textgreater\ dumpRaw( <integer> fileHandle\rightarrow )}
  ```

- **dumpXML**: Generates an XML dump of all properties.
  ```
  tfamily \textcolor{red}{\textless logical\textgreater\ dumpXML()}
  ```
16.5. Objects

tfamily double zero enclosedMass (double radius \rightarrow, componentType \rightarrow, massType \rightarrow, weightBy \rightarrow, integer \rightarrow, logical \rightarrow) Compute the mass enclosed within a radius.

tfamily red textless *type(treeNode) greater* host() Return a pointer to the host TreeNode object.

tfamily void hotHaloCoolingAbundancesRate (type(abundances) value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily void hotHaloCoolingAngularMomentumRate (double value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily void hotHaloCoolingMassRate (double value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily void initialize() Initialize the object.

tfamily void nameFromIndex (integer count \rightarrow, varying_string name \leftarrow) Return the name of a property given is index.

tfamily logical zero nullIsActive() Return whether the null implementation of the position component class is active.

tfamily void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily void output (integer integerProperty \leftrightarrow, integer integerBufferCount \leftrightarrow, integer[;] integerBuffer \leftrightarrow, integer doubleProperty \leftrightarrow, integer doubleBufferCount \leftrightarrow, double[;] doubleBuffer \leftrightarrow, double time \rightarrow, integer instance \rightarrow) Generate values of outputtable properties.

tfamily void outputCount (integer integerPropertyCount \leftrightarrow, integer doublePropertyCount \leftrightarrow, double time \rightarrow, integer instance \rightarrow) Compute a count of outputtable properties.

tfamily void outputNames (integer integerProperty \leftrightarrow, char[*](;) integerPropertyNames \leftrightarrow, char[*](;) integerPropertyComments \leftrightarrow, double[;] integerPropertyUnitsSI \leftrightarrow, integer doubleProperty \leftrightarrow, char[*](;) doublePropertyNames \leftrightarrow, char[*](;) doublePropertyComments \leftrightarrow, double[;] doublePropertyUnitsSI \leftrightarrow, double time \rightarrow, integer instance \rightarrow) Generate names of outputtable properties.

tfamily red textless double greater* position() Get the position property of the position component.

tfamily red textless type(varying_string)(;) greater positionAttributeMatch (logical [requireGettable], logical [requireSettable], logical [requireEvolvable]) Return a list of implementations that provide the given list off attributes for the position property of the position component.

tfamily red textless history greater positionHistory() Get the positionHistory property of the position component.

tfamily red textless type(history)(;) greater positionHistoryAttributeMatch (logical [requireGettable], logical [requireSettable], logical [requireEvolvable]) Return a list of implementations that provide the given list off attributes for the positionHistory property of the position component.
positionHistoryIsGettable() Get the positionHistory property of the position component.

tfamily \logicalzero positionHistoryIsSettable() Specify whether the positionHistory property of the position component is settable.

void positionHistorySet(<type(history)> value) Set the positionHistory property of the position component.

positionIsGettable() Get the position property of the position component.

positionIsSettable() Specify whether the position property of the position component is settable.

void positionSet(<double(:)> value) Set the position property of the position component.

potential(<double> radius \rightarrow componentType \rightarrow massType \rightarrow logical \rightarrow haloLoaded \rightarrow) Compute the gravitational potential.

presetIsActive() Return whether the preset implementation of the position component class is active.

readRaw(<integer> fileHandle \rightarrow) Read a binary dump of the nodeComponent from the given fileHandle.

rotationCurve(<double> radius \rightarrow componentType \rightarrow massType \rightarrow logical \rightarrow haloLoaded \rightarrow) Compute the rotation curve.

rotationCurveGradient(<double> radius \rightarrow componentType \rightarrow massType \rightarrow logical \rightarrow haloLoaded \rightarrow) Compute the rotation curve gradient.

serializeCount() Return a count of the number of evolvable quantities to be evolved.

serializeRates(<double(:)> array \rightarrow) Serialize the evolvable rates to an array.

serializeScales(<double(:)> array \rightarrow) Serialize the evolvable scales to an array.

serializeValues(<double(:)> array \rightarrow) Serialize the evolvable quantities to an array.

surfaceDensity(<double(3)> positionCylindrical \rightarrow componentType \rightarrow massType \rightarrow weightBy \rightarrow integer \rightarrow logical \rightarrow haloLoaded \rightarrow) Compute the surface density.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

tfamily \textcolor{red}{\textless double(:)\textgreater} velocity() Get the velocity property of the position component.

velocityAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the velocity property of the position component.

velocityIsGettable() Get the velocity property of the position component.
16.5. Objects

tfamily \logicalzero velocityIsSettable() Specify whether the velocity property of the position component is settable.

tfamily \void velocitySet(<double(:)> value) Set the velocity property of the position component.

tfamily nodeComponentSatellite

tfamily \textcolor{red}{\textless double\textgreater} boundMass() Get the boundMass property of the satellite component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)}\textgreater boundMassAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the boundMass property of the satellite component.

tfamily \intzero boundMassCount() Compute the count of evolvable quantities in the boundMass property of the SatelliteStandard component.

tfamily \textcolor{red}{\textlesstype(history)\textgreater} boundMassHistory() Get the boundMassHistory property of the satellite component.

tfamily \textcolor{red}{\textlesstype(history)\textgreater} boundMassHistoryAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the boundMassHistory property of the satellite component.

tfamily \logicalzero boundMassHistoryIsSettable() Specify whether the boundMassHistory property of the satellite component is settable.

tfamily \void boundMassHistorySet(<type(history)> value) Set the boundMassHistory property of the satellite component.

tfamily \textcolor{red}{\textlesstype(history)\textgreater} boundMassIsGettable() Get the boundMass property of the satellite component.

tfamily \logicalzero boundMassIsSettable() Specify whether the boundMass property of the satellite component is settable.

tfamily \void boundMassRate(<double> value) Cumulate to the rate of the boundMass property of the SatelliteStandard component.

tfamily \void boundMassScale(<double> value) Set the scale of the boundMass property of the SatelliteStandard component.

tfamily \void boundMassSet(<double> value) Set the boundMass property of the satellite component.

tfamily \void builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily \doublezero density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.
tfamily \textcolor{red}{\textless \textbf{double}():\textgreater} deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.
tfamily \textcolor{red}{\textless \textbf{double}():\textgreater} deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.
tfamily \void destroy() Destroy the object.
tfamily \void dump() Generate an ASCII dump of all properties.
tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.
tfamily \void dumpXML() Generate an XML dump of all properties.
tfamily \textcolor{red}{\textless \textbf{double}():\textgreater} enclosedMass(<double> radius→, <componentType>:componentType→, <massType>:massType→, <weightBy>:weightBy→, <integer>:weightIndex→, <logical>:haloLoaded→) Compute the mass enclosed within a radius.
tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.
tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.
tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.
tfamily \void initialize() Initialize the object.
tfamily \textcolor{red}{\textless \textbf{double}():\textgreater} mergeTime() Get the mergeTime property of the Satellite component.
tfamily \textcolor{red}{\textless \textbf{type(varying\_string):\textgreater}} mergeTimeAttributeMatch(<logical>:requireGettable→, <logical>:requireSettable→, <logical>:requireEvolvable→) Return a list of implementations that provide the given list off attributes for the mergeTime property of the Satellite component.
tfamily \int zero mergeTimeCount() Compute the count of evolvable quantities in the mergeTime property of the SatelliteVerySimple component.
tfamily \textcolor{red}{\textless \textbf{double}():\textgreater} mergeTimeIsGettable() Get the mergeTime property of the Satellite component.
tfamily \logical zero mergeTimeIsSettable() Specify whether the mergeTime property of the Satellite component is settable.
tfamily \void mergeTimeRate(<double> value) Cumulate to the rate of the mergeTime property of the SatelliteVerySimple component.
tfamily \void mergeTimeScale(<double> value) Set the scale of the mergeTime property of the SatelliteVerySimple component.
tfamily \void mergeTimeSet(<double> value) Set the mergeTime property of the Satellite component.
tfamily \void nameFromIndex(<integer> count→, <varying_string>:name←) Return the name of a property given is index.
16.5. Objects

tfamily \textcolor{red}{\langle\text{integer(kind=kind\_int8)}\rangle}\text{nodeIndex()}\text{Get the nodeIndex property of the satellite component.}

tfamily \textcolor{red}{\langle\text{type(varying\_string)}()\rangle}\text{nodeIndexAttributeMatch}(<\text{logical}>\text{	ext{[requireGettable]}},<\text{logical}>\text{	ext{[requireSettable]}},<\text{logical}>\text{	ext{[requireEvolvable]}})\text{Return a list of implementations that provide the given list off attributes for the nodeIndex property of the satellite component}

tfamily \textcolor{red}{\langle\text{type(longIntegerHistory)}\rangle}\text{nodeIndexHistory()}\text{Get the nodeIndexHistory property of the satellite component.}

tfamily \textcolor{red}{\langle\text{type(varying\_string)}()\rangle}\text{nodeIndexHistoryAttributeMatch(<\text{logical}>\text{	ext{[requireGettable]}},<\text{logical}>\text{	ext{[requireSettable]}},<\text{logical}>\text{	ext{[requireEvolvable]}})\text{Return a list of implementations that provide the given list off attributes for the nodeIndexHistory property of the satellite component}

tfamily \textcolor{red}{\langle\text{type(longIntegerHistory)}\rangle}\text{nodeIndexHistoryIsGettable()}\text{Get the nodeIndexHistory property of the satellite component.}

tfamily \textcolor{red}{\langle\text{type(longIntegerHistory)}\rangle}\text{nodeIndexHistoryIsSettable()}\text{Specify whether the nodeIndexHistory property of the satellite component is settable.}

tfamily \textcolor{red}{\langle\text{type(longIntegerHistory)}\rangle}\text{nodeIndexHistorySet(<\text{type(longIntegerHistory)}>}\text{value})\text{Set the nodeIndexHistory property of the satellite component.}

tfamily \textcolor{red}{\langle\text{integer(kind=kind\_int8)}\rangle}\text{nodeIsGettable()}\text{Get the nodeIndex property of the satellite component.}

tfamily \textcolor{red}{\langle\text{logical\_zero}\rangle}\text{nodeIsSettable()}\text{Specify whether the nodeIndex property of the satellite component is settable.}

tfamily \textcolor{red}{\langle\text{logical\_zero}\rangle}\text{nullIsActive()}\text{Return whether the null implementation of the satellite component class is active.}

tfamily \textcolor{red}{\langle\text{void}\rangle}\text{odeStepRatesInitialize()}\text{Initialize rates for evolvable properties.}

tfamily \textcolor{red}{\langle\text{void}\rangle}\text{odeStepScalesInitialize()}\text{Initialize scales for evolvable properties.}

tfamily \textcolor{red}{\langle\text{logical\_zero}\rangle}\text{orbitingIsActive()}\text{Return whether the orbiting implementation of the satellite component class is active.}

tfamily \textcolor{red}{\langle\text{void}\rangle}\text{output(<\text{integer}>\text{integerProperty}↔, <\text{integer}>\text{integerBufferCount}↔, <\text{integer}>\text{doubleProperty}↔, <\text{integer}>\text{doublePropertyCount}↔, <\text{double}>\text{doubleBuffer}↔, <\text{double}>\text{time}→, <\text{integer}>\text{instance}→)}\text{Generate values of outputtable properties.}

tfamily \textcolor{red}{\langle\text{void}\rangle}\text{outputCount(<\text{integer}>\text{integerPropertyCount}↔, <\text{integer}>\text{doublePropertyCount}↔, <\text{double}>\text{time}→, <\text{integer}>\text{instance}→)}\text{Compute a count of outputtable properties.}

tfamily \textcolor{red}{\langle\text{void}\rangle}\text{outputNames(<\text{integer}>\text{integerProperty}↔, <\text{char}[*]>\text{integerPropertyNames}↔, <\text{char}[*]>\text{integerPropertyComments}↔, <\text{double}()>\text{integerPropertyUnitsSI}↔, <\text{integer}>\text{doubleProperty}↔, <\text{char}[*]>\text{doublePropertyNames}↔, <\text{char}[*]>\text{doublePropertyComments}↔, <\text{double}()>\text{doublePropertyUnitsSI}↔, <\text{double}>\text{time}→, <\text{integer}>\text{instance}→)}\text{Generate names of outputtable properties.}

tfamily \textcolor{red}{\langle\text{double}\rangle}\text{position()}\text{Get the position property of the satellite component.}
16. Coding Galacticus

tfamily \textcolor{red}{\textless string\textgreater} positionAttributeMatch(\textcolor{red}{\textless logical\textgreater}[requireGettable] →, \textcolor{red}{\textless logical\textgreater}[requireSettable] →, \textcolor{red}{\textless logical\textgreater}[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the position property of the satellite component.

tfamily \intzero positionCount() Compute the count of evolvable quantities in the position property of the SatelliteOrbiting component.

tfamily \textcolor{red}{\textless double\textgreater} positionIsGettable() Get the position property of the satellite component.

tfamily \logicalzero positionIsSettable() Specify whether the position property of the satellite component is settable.

tfamily \void positionRate(\textcolor{red}{\textless double\textgreater} value) Cumulate to the rate of the position property of the SatelliteOrbiting component.

tfamily \void positionScale(\textcolor{red}{\textless double\textgreater} value) Set the scale of the position property of the SatelliteOrbiting component.

tfamily \void positionSet(\textcolor{red}{\textless double\textgreater} value) Set the position property of the satellite component.

tfamily \doublezero potential(\textcolor{red}{\textless double\textgreater} radius →, \textcolor{red}{\textless componentType\textgreater}[componentType] →, \textcolor{red}{\textless massType\textgreater}[massType] →, \textcolor{red}{\textless logical\textgreater}[haloLoaded] →) Compute the gravitational potential.

tfamily \logicalzero presetIsActive() Return whether the preset implementation of the satellite component class is active.

tfamily \doublezero rotationCurve(\textcolor{red}{\textless double\textgreater} radius →, \textcolor{red}{\textless componentType\textgreater}[componentType] →, \textcolor{red}{\textless massType\textgreater}[massType] →, \textcolor{red}{\textless logical\textgreater}[haloLoaded] →) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(\textcolor{red}{\textless double\textgreater} radius →, \textcolor{red}{\textless componentType\textgreater}[componentType] →, \textcolor{red}{\textless massType\textgreater}[massType] →, \textcolor{red}{\textless logical\textgreater}[haloLoaded] →) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(\textcolor{red}{\textless double\textgreater} array→) Serialize the evolvable rates to an array.

tfamily \void serializeScales(\textcolor{red}{\textless double\textgreater} array→) Serialize the evolvable scales to an array.

tfamily \void serializeValues(\textcolor{red}{\textless double\textgreater} array→) Serialize the evolvable quantities to an array.

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the satellite component class is active.

tfamily \doublezero surfaceDensity(\textcolor{red}{\textless double\textgreater}(3) positionCylindrical→, \textcolor{red}{\textless componentType\textgreater}[componentType]→, \textcolor{red}{\textless massType\textgreater}[massType]→, \textcolor{red}{\textless weightBy\textgreater}[weightBy]→, \textcolor{red}{\textless integer\textgreater}[weightIndex]→, \textcolor{red}{\textless logical\textgreater}[haloLoaded]→) Compute the surface density.

tfamily \textcolor{red}{\textless double\textgreater} tidalHeatingNormalized() Get the tidalHeatingNormalized property of the satellite component.

tfamily \textcolor{red}{\textless type\textgreater(varying\_string\_type}\textgreater} tidalHeatingNormalizedAttributeMatch(\textcolor{red}{\textless logical\textgreater}[requireGettable] →, \textcolor{red}{\textless logical\textgreater}[requireSettable] →, \textcolor{red}{\textless logical\textgreater}[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the tidalHeatingNormalized property of the satellite component.
tfamily\intzerotidalHeatingNormalizedCount() Compute the count of evolvable quantities in the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily\textcolor{red}{\textless\textgreater} tidalHeatingNormalizedIsGettable() Get the tidalHeatingNormalized property of the satellite component.

tfamily\logicalzero tidalHeatingNormalizedIsSettable() Specify whether the tidalHeatingNormalized property of the satellite component is settable.

tfamily\void tidalHeatingNormalizedRate(<double> value) Cumulate to the rate of the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily\void tidalHeatingNormalizedScale(<double> value) Set the scale of the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily\void tidalHeatingNormalizedSet(<double> value) Set the tidalHeatingNormalized property of the satellite component.

tfamily\textcolor{red}{\textless\textgreater} tidalTensorPathIntegrated() Get the tidalTensorPathIntegrated property of the satellite component.

tfamily\textcolor{red}{\textless \textgreater} tidalTensorPathIntegratedAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the tidalTensorPathIntegrated property of the satellite component.

tfamily\intzerotidalTensorPathIntegratedCount() Compute the count of evolvable quantities in the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily\textcolor{red}{\textless\textgreater} tidalTensorPathIntegratedIsGettable() Get the tidalTensorPathIntegrated property of the satellite component.

tfamily\logicalzero tidalTensorPathIntegratedIsSettable() Specify whether the tidalTensorPathIntegrated property of the satellite component is settable.

tfamily\void tidalTensorPathIntegratedRate(<type(tensorRank2Dimension3Symmetric)> value) Cumulate to the rate of the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily\void tidalTensorPathIntegratedScale(<type(tensorRank2Dimension3Symmetric)> value) Set the scale of the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily\void tidalTensorPathIntegratedSet(<type(tensorRank2Dimension3Symmetric)> value) Set the tidalTensorPathIntegrated property of the satellite component.

tfamily\textcolor{red}{\textless double\textgreater} timeOfMerging() Get the timeOfMerging property of the satellite component.

tfamily\textcolor{red}{\textless\textgreater} timeOfMergingAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the timeOfMerging property of the satellite component.

tfamily\textcolor{red}{\textless double\textgreater} timeOfMergingIsGettable() Get the timeOfMerging property of the satellite component.
tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

tfamily \textcolor{red}{\textless double(:)\textgreater} velocity() Get the velocity property of the satellite component.

tfamily \textcolor{red}{\textless type(keplerOrbit)\textgreater} virialOrbit() Get the virialOrbit property of the satellite component.

tfamily \textcolor{red}{\textless type(keplerOrbit)\textgreater} virialOrbitIsGettable() Get the virialOrbit property of the satellite component.

tfamily \textcolor{red}{\textless type(keplerOrbit)\textgreater} virialOrbitSet(\textcolor{red}{\textless type(keplerOrbit)\textgreater} value) Set the virialOrbit property of the satellite component.
16.5. Objects

tfamily nodeComponentSatelliteNull

tfamily \textcolor{red}{\textless \textasciitilde \textgreater } \text{boundMass}() Get the \textit{boundMass} property of the \textit{satellite} component.

tfamily \textcolor{red}{\textless \textasciitilde \textgreater } \text{boundMassAttributeMatch}(<\textit{logical}>[\text{requireGettable}→, <\textit{logical}>[\text{requireSettable}→, <\textit{logical}>[\text{requireEvolvable}→]) Return a list of implementations that provide the given list of attributes for the \textit{boundMass} property of the \textit{satellite} component.

tfamily \intzero \text{boundMassCount()} Compute the count of evolvable quantities in the \textit{boundMass} property of the \textit{SatelliteStandard} component.

tfamily \textcolor{red}{\textless \textasciitilde \textgreater } \text{boundMassHistory}() Get the \textit{boundMassHistory} property of the \textit{satellite} component.

tfamily \textcolor{red}{\textless \textasciitilde \textgreater } \text{boundMassHistoryAttributeMatch}(<\textit{logical}>[\text{requireGettable}→, <\textit{logical}>[\text{requireSettable}→, <\textit{logical}>[\text{requireEvolvable}→]) Return a list of implementations that provide the given list of attributes for the \textit{boundMassHistory} property of the \textit{satellite} component.

tfamily \textcolor{red}{\textless \textasciitilde \textgreater } \text{boundMassHistoryIsGettable()} Get the \textit{boundMassHistory} property of the \textit{satellite} component.

tfamily \textcolor{red}{\textless \textasciitilde \textgreater } \text{boundMassHistoryIsSettable()} Specify whether the \textit{boundMassHistory} property of the \textit{satellite} component is settable.

tfamily \void \text{boundMassHistorySet}(<\textit{type(history)}> value) Set the \textit{boundMassHistory} property of the \textit{satellite} component.

tfamily \textcolor{red}{\textless \textasciitilde \textgreater } \text{boundMassIsGettable()} Get the \textit{boundMass} property of the \textit{satellite} component.

tfamily \textcolor{red}{\textless \textasciitilde \textgreater } \text{boundMassIsSettable()} Specify whether the \textit{boundMass} property of the \textit{satellite} component is settable.

tfamily \void \text{boundMassRate}(<\textit{double}> value) Cumulate to the rate of the \textit{boundMass} property of the \textit{SatelliteStandard} component.

tfamily \void \text{boundMassScale}(<\textit{double}> value) Set the scale of the \textit{boundMass} property of the \textit{SatelliteStandard} component.

tfamily \void \text{boundMassSet}(<\textit{double}> value) Set the \textit{boundMass} property of the \textit{satellite} component.

tfamily \void \text{builder}(<\textit{type(node)}> componentDefinition→) Build a \textit{nodeComponent} from a supplied XML definition.

tfamily \void \text{density}(<\textit{double}(3)> positionSpherical→, <\textit{componentType}> [componentType]→, <\textit{massType}> [massType]→, <\textit{weightBy}> [weightBy]→, <\textit{integer}> [weightIndex]→, <\textit{logical}> [haloLoaded]→) Compute the density.

tfamily \void \text{deserializeRates}(<\textit{double}(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void \text{deserializeScales}(<\textit{double}(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void \text{deserializeValues}(<\textit{double}(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void \text{destroy}() Destroy the object.
16. Coding GALACTICUS

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \double zero enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}{\textless double\textgreater} mergeTime() Get the mergeTime property of the satellite component.

tfamily \textcolor{red}{\textless type(varying_string)()\textgreater} mergeTimeAttributeMatch(<logical> [requireGettable]→, <logical> [requireSettable]→, <logical> [requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the mergeTime property of the satellite component.

tfamily \int zero mergeTimeCount() Compute the count of evolvable quantities in the mergeTime property of the SatelliteVerySimple component.

tfamily \textcolor{red}{\textless double\textgreater} mergeTimeIsGettable() Get the mergeTime property of the satellite component.

tfamily \logical zero mergeTimeIsSettable() Specify whether the mergeTime property of the satellite component is settable.

tfamily \void mergeTimeRate(<double> value) Cumulate to the rate of the mergeTime property of the SatelliteVerySimple component.

tfamily \void mergeTimeScale(<double> value) Set the scale of the mergeTime property of the SatelliteVerySimple component.

tfamily \void mergeTimeSet(<double> value) Set the mergeTime property of the satellite component.

tfamily \void nameFromIndex(<integer> count→, <varying_string> name←) Return the name of a property given is index.

tfamily \textcolor{red}{\textless integer(kind=kind_int8)\textgreater} nodeIndex() Get the nodeIndex property of the satellite component.

tfamily \textcolor{red}{\textless type(varying_string)()\textgreater} nodeIndexAttributeMatch(<logical> [requireGettable]→, <logical> [requireSettable]→, <logical> [requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the nodeIndex property of the satellite component.
16.5. Objects

tfamily \textcolor{red}{\textless textless type(longIntegerHistory)\textgreater} nodeIndexHistory() Get the nodeIndexHistory property of the satellite component.

tfamily \textcolor{red}{\textless textless type(varying\textless String\textgreater)\textgreater} nodeIndexHistoryAttributeMatch(\textless logical\textgreater [requireGettable] →, \textless logical\textgreater [requireSettable] →, \textless logical\textgreater [requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the nodeIndexHistory property of the satellite component.

tfamily \textcolor{red}{\textless textless type(longIntegerHistory)\textgreater} nodeIndexHistoryIsGettable() Get the nodeIndexHistory property of the satellite component.

tfamily \textcolor{red}{\textless textless type(longIntegerHistory)\textgreater} nodeIndexHistoryIsSettable() Specify whether the nodeIndexHistory property of the satellite component is settable.

tfamily \textcolor{red}{\textless textless type(longIntegerHistory)\textgreater} nodeIndexHistorySet(\textless type(longIntegerHistory)\textgreater value) Set the nodeIndexHistory property of the satellite component.

tfamily \textcolor{red}{\textless integer(kind=kind\_int8)\textgreater} nodeIndexIsGettable() Get the nodeIndex property of the satellite component.

tfamily \textcolor{red}{\textless integer(kind=kind\_int8)\textgreater} nodeIndexIsSettable() Specify whether the nodeIndex property of the satellite component is settable.

tfamily \textcolor{red}{\textless integer(kind=kind\_int8)\textgreater} nullIsActive() Return whether the null implementation of the satellite component class is active.

tfamily \textcolor{red}{\textless integer(kind=kind\_int8)\textgreater} odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \textcolor{red}{\textless integer(kind=kind\_int8)\textgreater} odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \textcolor{red}{\textless integer(kind=kind\_int8)\textgreater} orbitingIsActive() Return whether the orbiting implementation of the satellite component class is active.

tfamily \textcolor{red}{\textless integer(kind=kind\_int8)\textgreater} output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

tfamily \textcolor{red}{\textless integer(kind=kind\_int8)\textgreater} outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily \textcolor{red}{\textless integer(kind=kind\_int8)\textgreater} outputNames(<integer> integerProperty ↔, <char\[*\](:)> integerPropertyNames ↔, <char\[*\](:)> integerPropertyComments ↔, <double(:)> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char\[*\](:)> doublePropertyNames ↔, <char\[*\](:)> doublePropertyComments ↔, <double(:)> doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

tfamily \textcolor{red}{\textless textless double\textless :\textgreater} position() Get the position property of the satellite component.

tfamily \textcolor{red}{\textless textless type(varying\textless String\textgreater)\textgreater} positionAttributeMatch(\textless logical\textgreater [requireGettable] →, \textless logical\textgreater [requireSettable] →, \textless logical\textgreater [requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the position property of the satellite component.

tfamily \textcolor{red}{\textless integer\textgreater} positionCount() Compute the count of evolvable quantities in the position property of the SatelliteOrbiting component.
16. Coding Galacticus

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater\textgreater} positionIsGettable() Get the position property of the satellite component.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} positionIsSettable() Specify whether the position property of the satellite component is settable.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} positionRate(<double(:)> value) Cumulate to the rate of the position property of the SatelliteOrbiting component.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} positionScale(<double(:)> value) Set the scale of the position property of the SatelliteOrbiting component.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} positionSet(<double(:)> value) Set the position property of the satellite component.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} potential(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the gravitational potential.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} presetIsActive() Return whether the preset implementation of the satellite component class is active.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} readRaw(<integer> fileHandle \rightarrow) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} rotationCurve(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the rotation curve.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} rotationCurveGradient(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the rotation curve gradient.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} serializeRates(<double(:)> array \rightarrow) Serialize the evolvable rates to an array.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} serializeScales(<double(:)> array \rightarrow) Serialize the evolvable scales to an array.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} serializeValues(<double(:)> array \rightarrow) Serialize the evolvable quantities to an array.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} standardIsActive() Return whether the standard implementation of the satellite component class is active.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} surfaceDensity(<double(3)> positionCylindrical \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <weightBy> [weightBy] \rightarrow, <integer> [weightIndex] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the surface density.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} tidalHeatingNormalized() Get the tidalHeatingNormalized property of the satellite component.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} tidalHeatingNormalizedAttributeMatch(<logical> [requireGettable] \rightarrow, <logical> [requireSettable] \rightarrow, <logical> [requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the tidalHeatingNormalized property of the satellite component.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} tidalHeatingNormalizedCount() Compute the count of evolvable quantities in the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily \textcolor{red}{\textless\textless\textless\textgreater\textgreater} tidalHeatingNormalizedIsGettable() Get the tidalHeatingNormalized property of the satellite component.
16.5. Objects

tfamily \logicalzero tidalHeatingNormalizedIsSettable() Specify whether the tidalHeatingNormalized property of the satellite component is settable.

tfamily \void tidalHeatingNormalizedRate(<double> value) Cumulate to the rate of the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily \void tidalHeatingNormalizedScale(<double> value) Set the scale of the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily \void tidalHeatingNormalizedSet(<double> value) Set the tidalHeatingNormalized property of the satellite component.

tfamily \void tidalHeatingNormalizedSet(<double> value) Set the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily \void tidalHeatingNormalizedSet(<double> value) Set the tidalHeatingNormalized property of the satellite component.

tfamily \void tidalHeatingNormalizedSet(<double> value) Set the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily \void tidalHeatingNormalizedSet(<double> value) Set the tidalHeatingNormalized property of the satellite component.

tfamily \void tidalHeatingNormalizedSet(<double> value) Set the tidalHeatingNormalized property of the satellite component.

tfamily \intzero tidalTensorPathIntegratedCount() Compute the count of evolvable quantities in the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily \void tidalTensorPathIntegratedRate(<type(tensorRank2Dimension3Symmetric)> value) Cumulate to the rate of the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily \void tidalTensorPathIntegratedScale(<type(tensorRank2Dimension3Symmetric)> value) Set the scale of the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily \void tidalTensorPathIntegratedSet(<type(tensorRank2Dimension3Symmetric)> value) Set the tidalTensorPathIntegrated property of the satellite component.

tfamily \void tidalTensorPathIntegratedSet(<double> value) Set the timeOfMerging property of the satellite component.

tfamily \void timeOfMergingSet(<double> value) Set the timeOfMerging property of the satellite component.
### Coding GALACTICUS

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type()</code></td>
<td>Return the type of the object.</td>
</tr>
<tr>
<td><code>velocity()</code></td>
<td>Get the velocity property of the satellite component.</td>
</tr>
<tr>
<td><code>velocityAttributeMatch(&lt;logical&gt;[requireGettable] → &lt;logical&gt;[requireSettable] → &lt;logical&gt;[requireEvolvable] →)</code></td>
<td>Return a list of implementations that provide the given list of attributes for the velocity property of the satellite component.</td>
</tr>
<tr>
<td><code>velocityCount()</code></td>
<td>Compute the count of evolvable quantities in the velocity property of the SatelliteOrbiting component.</td>
</tr>
<tr>
<td><code>velocityIsGettable()</code></td>
<td>Get the velocity property of the satellite component.</td>
</tr>
<tr>
<td><code>velocityIsSettable()</code></td>
<td>Specify whether the velocity property of the satellite component is settable.</td>
</tr>
<tr>
<td><code>velocityRate(&lt;double:]) value)</code></td>
<td>Cumulate to the rate of the velocity property of the SatelliteOrbiting component.</td>
</tr>
<tr>
<td><code>velocityScale(&lt;double:]) value)</code></td>
<td>Set the scale of the velocity property of the SatelliteOrbiting component.</td>
</tr>
<tr>
<td><code>velocitySet(&lt;double:]) value)</code></td>
<td>Set the velocity property of the satellite component.</td>
</tr>
<tr>
<td><code>verySimpleIsActive()</code></td>
<td>Return whether the verySimple implementation of the satellite component class is active.</td>
</tr>
<tr>
<td><code>virialOrbit()</code></td>
<td>Get the virialOrbit property of the satellite component.</td>
</tr>
<tr>
<td><code>virialOrbitAttributeMatch(&lt;logical&gt;[requireGettable] → &lt;logical&gt;[requireSettable] → &lt;logical&gt;[requireEvolvable] →)</code></td>
<td>Return a list of implementations that provide the given list of attributes for the virialOrbit property of the satellite component.</td>
</tr>
<tr>
<td><code>virialOrbitIsGettable()</code></td>
<td>Get the virialOrbit property of the satellite component.</td>
</tr>
<tr>
<td><code>virialOrbitIsSettable()</code></td>
<td>Specify whether the virialOrbit property of the satellite component is settable.</td>
</tr>
<tr>
<td><code>virialOrbitSet(&lt;type(keplerOrbit)&gt; value)</code></td>
<td>Set the virialOrbit property of the satellite component.</td>
</tr>
<tr>
<td><code>boundMass()</code></td>
<td>Get the boundMass property of the satellite component.</td>
</tr>
<tr>
<td><code>boundMassAttributeMatch(&lt;logical&gt;[requireGettable] → &lt;logical&gt;[requireSettable] → &lt;logical&gt;[requireEvolvable] →)</code></td>
<td>Return a list of implementations that provide the given list of attributes for the boundMass property of the satellite component.</td>
</tr>
<tr>
<td><code>boundMassSet(&lt;type(keplerOrbit)&gt; value)</code></td>
<td>Set the boundMass property of the satellite component.</td>
</tr>
</tbody>
</table>
16.5. Objects

```plaintext
tfamily \intzero boundMassCount() Compute the count of evolvable quantities in the boundMass property of the SatelliteStandard component.

tfamily \textcolor{red}{\textless type(history)}\textgreater boundMassHistory() Get the boundMassHistory property of the satellite component.

tfamily \textcolor{red}{\textless type(varying\_string)}\textgreater boundMassHistoryAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the boundMassHistory property of the satellite component.

tfamily \textcolor{red}{\textless type(history)}\textgreater boundMassHistoryIsGettable() Get the boundMassHistory property of the satellite component.

tfamily \logicalzero boundMassHistoryIsSettable() Specify whether the boundMassHistory property of the satellite component is settable.

tfamily \void boundMassHistorySet(<type(history)> value) Set the boundMassHistory property of the satellite component.

tfamily \textcolor{red}{\textless double}\textgreater boundMassIsGettable() Get the boundMass property of the satellite component.

tfamily \logicalzero boundMassIsSettable() Specify whether the boundMass property of the satellite component is settable.

tfamily \void boundMassRate(<double> value) Cumulate to the rate of the boundMass property of the SatelliteStandard component.

tfamily \void boundMassScale(<double>value) Set the scale of the boundMass property of the SatelliteStandard component.

tfamily \void boundMassSet(<double> value) Set the boundMass property of the satellite component.

tfamily \void builder(<\type(node)>componentDefinition) Build a nodeComponent from a supplied XML definition.

tfamily \void density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.
```
tfamily doublezero enclosedMass(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <weightBy> [weightBy] \rightarrow, <integer> [weightIndex] \rightarrow, <logical> [haloLoaded] \rightarrow)
Compute the mass enclosed within a radius.

tfamily textcolor{red}{'textless *type(treeNode)\textgreater} host() Return a pointer to the host
   treeNode object.

tfamily void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the
   hotHaloCoolingAbundances property of the hotHalo component.

tfamily void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the
   hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass
   property of the hotHalo component.

tfamily void initialize() Initialize the object.

tfamily textcolor{red}{'textless double\textgreater} mergeTime() Get the mergeTime property of the
   satellite component.

tfamily textcolor{red}{'textless type(varying\_string)\textgreater} mergeTimeAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that
   provide the given list off attributes for the mergeTime property of the satellite component

tfamily intzero mergeTimeCount() Compute the count of evolvable quantities in the mergeTime prop-
   erty of the SatelliteVerySimple component.

tfamily textcolor{red}{'textless double\textgreater} mergeTimeIsGettable() Get the mergeTime prop-
   erty of the satellite component.

tfamily logicalzero mergeTimeIsSettable() Specify whether the mergeTime property of the satellite
   component is settable.

tfamily void mergeTimeRate(<double> value) Cumulate to the rate of the mergeTime property of the
   SatelliteVerySimple component.

tfamily void mergeTimeSet(<double> value) Set the mergeTime property of the SatelliteVerySimple
   component.

tfamily void nameFromIndex(<integer> count \rightarrow, <varying_string>name \leftarrow) Return the name of a
   property given is index.

tfamily textcolor{red}{'textless integer(kind=kind\_int8)\textgreater} nodeIndex() Get the nodeIndex
   property of the satellite component.

tfamily textcolor{red}{'textless type(varying\_string)\textgreater} nodeIndexAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that
   provide the given list off attributes for the nodeIndex property of the satellite component

tfamily textcolor{red}{'textless type(longIntegerHistory)\textgreater} nodeIndexHistory() Get the nodeIndexHistory
   property of the satellite component.
16.5. Objects

tfamily \textcolor{red}{\textless type(varying\_string)(:\textgreater)} nodeIndexHistoryAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the nodeIndexHistory property of the satellite component.

tfamily \textcolor{red}{\textless type(longIntegerHistory)\textgreater} nodeIndexHistoryIsGettable() Get the nodeIndexHistory property of the satellite component.

tfamily \logicalzero nodeIndexHistoryIsSettable() Specify whether the nodeIndexHistory property of the satellite component is settable.

tfamily \void nodeIndexHistorySet(<type(longIntegerHistory)> value) Set the nodeIndexHistory property of the satellite component.

tfamily \logicalzero nodeIndexIsGettable() Get the nodeIndex property of the satellite component.

tfamily \logicalzero nodeIndexIsSettable() Specify whether the nodeIndex property of the satellite component is settable.

tfamily \void nullIsActive() Return whether the null implementation of the satellite component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \logicalzero orbitingIsActive() Return whether the orbiting implementation of the satellite component class is active.

tfamily \void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(_:,:)> integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(_:,:)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

tfamily \void outputCount(<integer>integerPropertyCount ↔, <integer>doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily \void outputNames(<integer>integerProperty ↔, <char[*](:)>integerPropertyNames ↔, <char[*](:)>integerPropertyComments ↔, <double(:)>integerPropertyUnitsSI ↔, <integer>doubleProperty ↔, <char[*](:)>doublePropertyNames ↔, <char[*](:)>doublePropertyComments ↔, <double(:)>doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

tfamily \textcolor{red}{\textless double(:)\textgreater} position() Get the position property of the satellite component.

tfamily \textcolor{red}{\textless type(varying\_string)(:\textgreater)} positionAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the position property of the satellite component.

tfamily \textcolor{red}{\textless type(longIntegerHistory)\textgreater} positionIsGettable() Get the position property of the satellite component.
16. **Coding Galacticus**

tfamily `logical zero positionIsSettable()` Specify whether the `position` property of the `satellite` component is settable.

tfamily `void positionRate(<double(:)> value)` Cumulate to the rate of the `position` property of the `SatelliteOrbiting` component.

tfamily `void positionScale(<double(:)> value)` Set the scale of the `position` property of the `SatelliteOrbiting` component.

tfamily `void positionSet(<double(:)> value)` Set the `position` property of the `satellite` component.

tfamily `double zero potential(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow)` Compute the gravitational potential.

tfamily `logical zero presetIsActive()` Return whether the preset implementation of the satellite component class is active.

tfamily `void readRaw(<integer> fileHandle \rightarrow)` Read a binary dump of the `nodeComponent` from the given `fileHandle`.

tfamily `double zero rotationCurve(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow)` Compute the rotation curve.

tfamily `double zero rotationCurveGradient(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow)` Compute the rotation curve gradient.

tfamily `int zero serializeCount()` Return a count of the number of evolvable quantities to be evolved.

tfamily `void serializeRates(<double(:)> array \rightarrow)` Serialize the evolvable rates to an array.

tfamily `void serializeScales(<double(:)> array \rightarrow)` Serialize the evolvable scales to an array.

tfamily `void serializeValues(<double(:)> array \rightarrow)` Serialize the evolvable quantities to an array.

tfamily `logical zero standardIsActive()` Return whether the standard implementation of the satellite component class is active.

tfamily `double zero surfaceDensity(<double(3)> positionCylindrical \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <weightBy> [weightBy] \rightarrow, <integer> [weightIndex] \rightarrow, <logical> [haloLoaded] \rightarrow)` Compute the surface density.

tfamily `textcolor {red} {textless double textgreater} tidalHeatingNormalized()` Get the `tidalHeatingNormalized` property of the `satellite` component.

tfamily `textcolor {red} {textless varying \_string(): textgreater} tidalHeatingNormalizedAttributeMatch([<logical> [requireGettable] \rightarrow, <logical> [requireSettable] \rightarrow, <logical> [requireEvolvable] \rightarrow)` Return a list of implementations that provide the given list of attributes for the `tidalHeatingNormalized` property of the `satellite` component.

tfamily `int zero tidalHeatingNormalizedCount()` Compute the count of evolvable quantities in the `tidalHeatingNormalized` property of the `SatelliteOrbiting` component.

tfamily `textcolor {red} {textless double textgreater} tidalHeatingNormalizedIsGettable()` Get the `tidalHeatingNormalized` property of the `satellite` component.

tfamily `logical zero tidalHeatingNormalizedIsSettable()` Specify whether the `tidalHeatingNormalized` property of the `satellite` component is settable.
16.5. Objects

tfamily \void tidalHeatingNormalizedRate(\textsf{double} value) Cumulate to the rate of the \texttt{tidalHeatingNormalized} property of the \texttt{SatelliteOrbiting} component.

tfamily \void tidalHeatingNormalizedScale(\textsf{double} value) Set the scale of the \texttt{tidalHeatingNormalized} property of the \texttt{SatelliteOrbiting} component.

tfamily \void tidalHeatingNormalizedSet(\textsf{double} value) Set the \texttt{tidalHeatingNormalized} property of the \texttt{Satellite} component.

tfamily \textcolor{red}{\textlesstype(tensorRank2Dimension3Symmetric)\textgreater} tidalTensorPathIntegrated() Get the \texttt{tidalTensorPathIntegrated} property of the \texttt{Satellite} component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)} tidalTensorPathIntegratedAttributeMatch(\textsf{logical} \rightarrow, \textsf{logical} \rightarrow, \textsf{logical} \rightarrow) Return a list of implementations that provide the given list of attributes for the \texttt{tidalTensorPathIntegrated} property of the \texttt{Satellite} component.

tfamily \intzero tidalTensorPathIntegratedCount() Compute the count of evolvable quantities in the \texttt{tidalTensorPathIntegrated} property of the \texttt{SatelliteOrbiting} component.

tfamily \textcolor{red}{\textlesstype(tensorRank2Dimension3Symmetric)\textgreater} tidalTensorPathIntegratedIsGettable() Get the \texttt{tidalTensorPathIntegrated} property of the \texttt{Satellite} component.

tfamily \logicalzero tidalTensorPathIntegratedIsSettable() Specify whether the \texttt{tidalTensorPathIntegrated} property of the \texttt{Satellite} component is settable.

tfamily \void tidalTensorPathIntegratedRate(\textsf{type(tensorRank2Dimension3Symmetric)} value) Cumulate to the rate of the \texttt{tidalTensorPathIntegrated} property of the \texttt{SatelliteOrbiting} component.

tfamily \void tidalTensorPathIntegratedScale(\textsf{type(tensorRank2Dimension3Symmetric)} value) Set the scale of the \texttt{tidalTensorPathIntegrated} property of the \texttt{SatelliteOrbiting} component.

tfamily \void tidalTensorPathIntegratedSet(\textsf{type(tensorRank2Dimension3Symmetric)} value) Set the \texttt{tidalTensorPathIntegrated} property of the \texttt{Satellite} component.

tfamily \textcolor{red}{\textlesstype(varying\_string)\textgreater} timeOfMerging() Get the \texttt{timeOfMerging} property of the \texttt{Satellite} component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)} timeOfMergingAttributeMatch(\textsf{logical} \rightarrow, \textsf{logical} \rightarrow, \textsf{logical} \rightarrow) Return a list of implementations that provide the given list of attributes for the \texttt{timeOfMerging} property of the \texttt{Satellite} component.

tfamily \textcolor{red}{\textlesstype(varying\_string)\textgreater} timeOfMergingIsGettable() Get the \texttt{timeOfMerging} property of the \texttt{Satellite} component.

tfamily \logicalzero timeOfMergingIsSettable() Specify whether the \texttt{timeOfMerging} property of the \texttt{Satellite} component is settable.

tfamily \void timeOfMergingSet(\textsf{double} value) Set the \texttt{timeOfMerging} property of the \texttt{Satellite} component.

tfamily \textcolor{red}{\textlesstype(varying\_string)\textgreater} type() Return the type of this object.
16. Coding GALACTICUS

tfamily \textcolor{red}{\textless double(:)\textgreater} velocity() Get the velocity property of the satellite component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} velocityAttributeMatch(<logical>[requireGettable]\rightarrow,<logical>[requireSettable]\rightarrow,<logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list off attributes for the velocity property of the satellite component.

tfamily \intzero velocityCount() Compute the count of evolvable quantities in the velocity property of the SatelliteOrbiting component.

tfamily \textcolor{red}{\textless double(:)\textgreater} velocityIsGettable() Get the velocity property of the satellite component.

tfamily \logicalzero velocityIsSettable() Specify whether the velocity property of the satellite component is settable.

tfamily void velocityRate(<double(:)> value) Cumulate to the rate of the velocity property of the SatelliteOrbiting component.

tfamily void velocityScale(<double(:)> value) Set the scale of the velocity property of the SatelliteOrbiting component.

tfamily \logicalzero verySimpleIsActive() Return whether the verySimple implementation of the satellite component class is active.

tfamily \textcolor{red}{\textless type(keplerOrbit)\textgreater} virialOrbit() Get the virialOrbit property of the satellite component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} virialOrbitAttributeMatch(<logical>[requireGettable]\rightarrow,<logical>[requireSettable]\rightarrow,<logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list off attributes for the virialOrbit property of the satellite component.

tfamily \textcolor{red}{\textless type(keplerOrbit)\textgreater} virialOrbitIsGettable() Get the virialOrbit property of the satellite component.

tfamily \logicalzero virialOrbitIsSettable() Specify whether the virialOrbit property of the satellite component is settable.

tfamily \void virialOrbitSet(<type(keplerOrbit)> value) Set the virialOrbit property of the satellite component.

tfamily nodeComponentSatellitePreset

tfamily \textcolor{red}{\textless double(:)\textgreater} boundMass() Get the boundMass property of the satellite component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} boundMassAttributeMatch(<logical>[requireGettable]\rightarrow,<logical>[requireSettable]\rightarrow,<logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list off attributes for the boundMass property of the satellite component.

tfamily \intzero boundMassCount() Compute the count of evolvable quantities in the boundMass property of the SatelliteStandard component.
16.5. Objects

tfamily {textcolor{red}\{textless type(history)\textgreater}} boundMassHistory() Get the boundMassHistory property of the satellite component.

tfamily {textcolor{red}\{textless type(varying_string)\textgreater}} boundMassHistoryAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the boundMassHistory property of the satellite component.

tfamily {textcolor{red}\{textless type(history)\textgreater}} boundMassHistoryIsGettable() Get the boundMassHistory property of the satellite component.

tfamily {logicalzero} boundMassHistoryIsSettable() Specify whether the boundMassHistory property of the satellite component is settable.

tfamily {void} boundMassHistorySet(<type(history)> value) Set the boundMassHistory property of the satellite component.

tfamily {textcolor{red}\{textless double\textgreater}} boundMassIsGettable() Get the boundMass property of the satellite component.

tfamily {logicalzero} boundMassIsSettable() Specify whether the boundMass property of the satellite component is settable.

tfamily {void} boundMassRate(<double> value) Cumulate to the rate of the boundMass property of the SatelliteStandard component.

tfamily {void} boundMassScale(<double> value) Set the scale of the boundMass property of the SatelliteStandard component.

tfamily {void} boundMassSet(<double> value) Set the boundMass property of the satellite component.

tfamily {void} builder(<*type(node)> componentDefinition →) Build a nodeComponent from a supplied XML definition.

tfamily {doublezero} density(<double(3)> positionSpherical →, <componentType>[componentType] →, <massType>[massType] →, <weightBy>[weightBy] →, <integer>[weightIndex] →, <logical>[haloLoaded] →) Compute the density.

tfamily {void} deserializeRates(<double(:)> array ←) Deserialize the evolvable rates from an array.

tfamily {void} deserializeScales(<double(:)> array ←) Deserialize the evolvable scales from an array.

tfamily {void} deserializeValues(<double(:)> array ←) Deserialize the evolvable quantities from an array.

tfamily {void} destroy() Destroy the object.

tfamily {void} dump() Generate an ASCII dump of all properties.

tfamily {void} dumpRaw(<integer> fileHandle →) Generate a binary dump of all properties.

tfamily {void} dumpXML() Generate an XML dump of all properties.

tfamily {doublezero} enclosedMass(<double> radius →, <componentType>[componentType] →, <massType>[massType] →, <weightBy>[weightBy] →, <integer>[weightIndex] →, <logical>[haloLoaded] →) Compute the mass enclosed within a radius.
16. Coding Galacticus

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host
\textcolor{red}{\textless type(abundances)\textgreater} hotHaloCoolingAbundancesRate() Cumulate to the rate of the
hotHaloCoolingAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} hotHaloCoolingAngularMomentumRate() Cumulate to the rate of the
hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} hotHaloCoolingMassRate() Cumulate to the rate of the hotHaloCoolingMass
property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}{\textless double\textgreater} mergeTime() Get the mergeTime property of the
\textcolor{red}{\textless varying\_string\textgreater	extless:	extgreater} mergeTimeAttributeMatch() Return a list of implementations that
provide the given list off attributes for the mergeTime property of the satellite component

tfamily \intzero mergeTimeCount() Compute the count of evolvable quantities in the mergeTime property of the SatelliteVerySimple component.

tfamily \textcolor{red}{\textless double\textgreater} mergeTimeIsGettable() Get the mergeTime property of the
\textcolor{red}{\textlesstype(longIntegerHistory)\textgreater} mergeTimeScale() Set the scale of the mergeTime property of the SatelliteVerySimple component.

tfamily \void mergeTimeSet() Set the mergeTime property of the satellite component.

tfamily \void nameFromIndex() Specify whether the mergeTime property of the satellite component is settable.

tfamily \void mergeTimeRate() Cumulate to the rate of the mergeTime property of the SatelliteVerySimple component.

tfamily \void nodeIndex() Get the nodeIndex property of the satellite component.

tfamily \textcolor{red}{\textless varying\_string\textgreater} nodeIndexAttributeMatch() Return a list of implementations that
provide the given list off attributes for the nodeIndex property of the satellite component

tfamily \textcolor{red}{\textlesstype(longIntegerHistory)\textgreater} nodeIndexHistory() Get the nodeIndexHistory property of the satellite component.
16.5. Objects

tfamily \textcolor{red}{\textless type(longIntegerHistory)\textgreater} nodeIndexHistoryIsGettable() Get the nodeIndexHistory property of the satellite component.

tfamily \logicalzero nodeIndexHistoryIsSettable() Specify whether the nodeIndexHistory property of the satellite component is settable.

tfamily \void nodeIndexHistorySet(<type(longIntegerHistory)> value) Set the nodeIndexHistory property of the satellite component.

tfamily \textcolor{red}{\textless integer(kind=kind\_int8)\textgreater} nodeIndexIsGettable() Get the nodeIndex property of the satellite component.

tfamily \logicalzero nodeIndexIsSettable() Specify whether the nodeIndex property of the satellite component is settable.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the satellite component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \logicalzero orbitingIsActive() Return whether the orbiting implementation of the satellite component class is active.

tfamily \void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(;)> integerBuffer ↔, <integer> doubleProperty ↔, <integer> doubleBufferCount ↔, <double(;)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty ↔, <char[*]()> integerPropertyNames ↔, <char[*]()> integerPropertyComments ↔, <double(;)> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char[*]()> doublePropertyNames ↔, <char[*]()> doublePropertyComments ↔, <double(;)> doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

16.5. Objects

tfamily \textcolor{red}{\textless double(;)\textgreater} position() Get the position property of the satellite component.

tfamily \textcolor{red}{\textless type(varying\_string)(;)\textgreater} positionAttributeMatch(<logical>[requireGettable]->, <logical>[requireSettable]->, <logical>[requireEvolvable]->) Return a list of implementations that provide the given list off attributes for the position property of the satellite component.

tfamily \intzero positionCount() Compute the count of evolvable quantities in the position property of the SatelliteOrbiting component.

tfamily \textcolor{red}{\textless double(;)\textgreater} positionIsGettable() Get the position property of the satellite component.

tfamily \logicalzero positionIsSettable() Specify whether the position property of the satellite component is settable.

tfamily \void positionRate(<double(;)> value) Cumulate to the rate of the position property of the SatelliteOrbiting component.
tfamily \ void positionScale\textless double\textgreater\ value) Set the scale of the position property of the SatelliteOrbiting component.

tfamily \ void positionSet\textless double\textgreater\ value) Set the position property of the satellite component.

tfamily \ double zero potential\textless double\textgreater\ radius→, \ componentType\textgreater\ [componentType]→, \ massType\textgreater\ [massType]→, \ logical\ [haloLoaded]→) Compute the gravitational potential.

tfamily \ logical zero presetIsActive() Return whether the preset implementation of the satellite component class is active.

tfamily \ void readRaw\textless integer\textgreater\ fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \ double zero rotationCurve\textless double\textgreater\ radius→, \ componentType\textgreater\ [componentType]→, \ massType\textgreater\ [massType]→, \ logical\ [haloLoaded]→) Compute the rotation curve.

tfamily \ double zero rotationCurveGradient\textless double\textgreater\ radius→, \ componentType\textgreater\ [componentType]→, \ massType\textgreater\ [massType]→, \ logical\ [haloLoaded]→) Compute the rotation curve gradient.

tfamily \ int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \ void serializeRates\textless double\textgreater\ array→) Serialize the evolvable rates to an array.

tfamily \ void serializeScales\textless double\textgreater\ array→) Serialize the evolvable scales to an array.

tfamily \ void serializeValues\textless double\textgreater\ array→) Serialize the evolvable quantities to an array.

tfamily \ logical zero standardIsActive() Return whether the standard implementation of the satellite component class is active.

tfamily \ double zero surfaceDensity\textless double(3)\textgreater\ positionCylindrical→, \ componentType\textgreater\ [componentType]→, \ massType\textgreater\ [massType]→, \ weightBy\textless string\textgreater\ [weightBy]→, \ integer\ [weightIndex]→, \ logical\ [haloLoaded]→) Compute the surface density.

tfamily \ double \textcolor{red}{\textless double\textgreater} tidalHeatingNormalized() Get the tidalHeatingNormalized property of the satellite component.

tfamily \ double \textcolor{red}{\textless double\textgreater\ type(varying_string):} tidalHeatingNormalizedAttributeMatch\textless logical\ [requireGettable]→, \ logical\ [requireSettable]→, \ logical\ [requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the tidalHeatingNormalized property of the satellite component

tfamily \ int zero tidalHeatingNormalizedCount() Compute the count of evolvable quantities in the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily \ double \textcolor{red}{\textless double\textgreater\ type} tidalHeatingNormalizedIsGettable() Get the tidalHeatingNormalized property of the satellite component.

tfamily \ logical zero tidalHeatingNormalizedIsSettable() Specify whether the tidalHeatingNormalized property of the satellite component is settable.

tfamily \ void tidalHeatingNormalizedRate\textless double\textgreater\ value) Cumulate to the rate of the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily \ void tidalHeatingNormalizedScale\textless double\textgreater\ value) Set the scale of the tidalHeatingNormalized property of the SatelliteOrbiting component.
16.5. Objects

tfamily \void tidalHeatingNormalizedSet(<double> value) Set the tidalHeatingNormalized property of the satellite component.

tfamily \textcolor{red}{\textless\text{textless type(tensorRank2Dimension3Symmetric)\textgreater}} tidalTensorPathIntegrated() Get the tidalTensorPathIntegrated property of the satellite component.

tfamily \textcolor{red}{\textless\text{textless type(varying\_string)\textgreater}} tidalTensorPathIntegratedAttributeMatch(<logical>[requireGettable] → , <logical>[requireSettable] → , <logical>[requireEvolvable] → ) Return a list of implementations that provide the given list of attributes for the tidalTensorPathIntegrated property of the satellite component.

tfamily \intzero tidalTensorPathIntegratedCount() Compute the count of evolvable quantities in the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily \textcolor{red}{\textless\text{textless type(tensorRank2Dimension3Symmetric)\textgreater}} tidalTensorPathIntegratedIsGettable() Get the tidalTensorPathIntegrated property of the satellite component.

tfamily \logicalzero tidalTensorPathIntegratedIsSettable() Specify whether the tidalTensorPathIntegrated property of the satellite component is settable.

tfamily \void tidalTensorPathIntegratedRate(<type(tensorRank2Dimension3Symmetric)> value) Cumulate to the rate of the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily \void tidalTensorPathIntegratedScale(<type(tensorRank2Dimension3Symmetric)> value) Set the scale of the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily \void tidalTensorPathIntegratedSet(<type(tensorRank2Dimension3Symmetric)> value) Set the tidalTensorPathIntegrated property of the satellite component.

tfamily \\textcolor{red}{\textless\text{type(varying\_string)\textgreater}} type() Return the type of this object.

tfamily \\textcolor{red}{\textless\text{textless type(varying\_string)\textgreater}} velocity() Get the velocity property of the satellite component.

tfamily \\textcolor{red}{\textless\text{textless type(varying\_string)\textgreater}} velocityAttributeMatch(<logical>[requireGettable] → , <logical>[requireSettable] → , <logical>[requireEvolvable] → ) Return a list of implementations that provide the given list of attributes for the velocity property of the satellite component.
tfamily `\intzero` velocityCount() Compute the count of evolvable quantities in the `velocity` property of the `SatelliteOrbiting` component.

tfamily `\textcolor{red}{\textless double(:)\textgreater}` velocityIsGettable() Get the `velocity` property of the `satellite` component.

tfamily `\logicalzero` velocityIsSettable() Specify whether the `velocity` property of the `satellite` component is settable.

tfamily `\void` velocityRate(<`double(:)`> value) Cumulate to the rate of the `velocity` property of the `SatelliteOrbiting` component.

tfamily `\void` velocityScale(<`double(:)`> value) Set the scale of the `velocity` property of the `SatelliteOrbiting` component.

tfamily `\void` velocitySet(<`double(:)`> value) Set the `velocity` property of the `satellite` component.

tfamily `\logicalzero` verySimpleIsActive() Return whether the verySimple implementation of the `satellite` component class is active.

tfamily `\textcolor{red}{\textless type(keplerOrbit)\textgreater}` virialOrbit() Get the `virialOrbit` property of the `satellite` component.

tfamily `\textcolor{red}{\textlesstype(varying_string)(:)\textgreater}` virialOrbitAttributeMatch(<`logical`>[requireGettable]→, <`logical`>[requireSettable]→, <`logical`>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the `virialOrbit` property of the `satellite` component

tfamily `\textcolor{red}{\textlesstype(keplerOrbit)\textgreater}` virialOrbitIsGettable() Get the `virialOrbit` property of the `satellite` component.

tfamily `\logicalzero` virialOrbitIsSettable() Specify whether the `virialOrbit` property of the `satellite` component is settable.

tfamily `\void` virialOrbitSet(<`type(keplerOrbit)`> value) Set the `virialOrbit` property of the `satellite` component.

tfamily `\textcolor{red}{\textless double\textgreater}` boundMass() Get the `boundMass` property of the `satellite` component.

tfamily `\textcolor{red}{\textlesstype(varying_string)(:)\textgreater}` boundMassAttributeMatch(<`logical`>[requireGettable]→, <`logical`>[requireSettable]→, <`logical`>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the `boundMass` property of the `satellite` component

tfamily `\intzero` boundMassCount() Compute the count of evolvable quantities in the `boundMass` property of the `SatelliteStandard` component.

tfamily `\textcolor{red}{\textlesstype(history)(:)\textgreater}` boundMassHistory() Get the `boundMassHistory` property of the `satellite` component.

tfamily `\textcolor{red}{\textlesstype(varying_string)(:)\textgreater}` boundMassHistoryAttributeMatch(<`logical`>[requireGettable]→, <`logical`>[requireSettable]→, <`logical`>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the `boundMassHistory` property of the `satellite` component
16.5. Objects

tfamily \textcolor{red}{\textless type(history)\textgreater} boundMassHistoryIsGettable() Get the boundMassHistory property of the satellite component.

tfamily \logicalzero boundMassHistoryIsSettable() Specify whether the boundMassHistory property of the satellite component is settable.

tfamily \void boundMassHistorySet(<type(history)> value) Set the boundMassHistory property of the satellite component.

tfamily \textcolor{red}{\textless double\textgreater} boundMassIsGettable() Get the boundMass property of the satellite component.

tfamily \logicalzero boundMassIsSettable() Specify whether the boundMass property of the satellite component is settable.

tfamily \void boundMassRate(<double> value) Cumulate to the rate of the boundMass property of the SatelliteStandard component.

tfamily \void boundMassScale(<double> value) Set the scale of the boundMass property of the SatelliteStandard component.

tfamily \void boundMassSet(<double> value) Set the boundMass property of the satellite component.

tfamily \void builder(<type(node)>componentDefinition →) Build a nodeComponent from a supplied XML definition.

tfamily \doublezero density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless type(treeNode)\textgreater} host() Return a pointer to the host TreeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.
tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}{\textless double\textgreater} mergeTime() Get the mergeTime property of the satellite component.

tfamily \textcolor{red}{\textless double\textgreater} mergeTime() Get the mergeTime property of the satellite component.

tfamily \textcolor{red}{\textless double\textgreater} mergeTimeIsGettable() Get the mergeTime property of the satellite component.

tfamily \logical zero mergeTimeIsSettable() Specify whether the mergeTime property of the satellite component is settable.

tfamily \void mergeTimeRate(<double> value) Cumulate to the rate of the mergeTime property of the SatelliteVerySimple component.

tfamily \void mergeTimeScale(<double> value) Set the scale of the mergeTime property of the SatelliteVerySimple component.

tfamily \void mergeTimeSet(<double> value) Set the mergeTime property of the satellite component.

tfamily \void nameFromIndex(<integer> count →, <varying_string> name ←) Return the name of a property given is index.

tfamily \textcolor{red}{\textless integer(kind=kind_int8)\textgreater} nodeIndex() Get the nodeIndex property of the satellite component.

tfamily \textcolor{red}{\textless type(varying_string)\textgreater} nodeIndexHistory() Get the nodeIndexHistory property of the satellite component.

tfamily \textcolor{red}{\textless type(longIntegerHistory)\textgreater} nodeIndexHistory() Get the nodeIndexHistory property of the satellite component.

tfamily \textcolor{red}{\textless type(longIntegerHistory)\textgreater} nodeIndexHistoryIsGettable() Get the nodeIndexHistory property of the satellite component.

tfamily \logical zero nodeIndexHistoryIsSettable() Specify whether the nodeIndexHistory property of the satellite component is settable.

tfamily \void nodeIndexHistorySet(<type(longIntegerHistory)> value) Set the nodeIndexHistory property of the satellite component.
16.5. Objects

tfamily \textcolor{red}{\textless integer(kind=kind\_int8)\textgreater} nodeIndexIsGettable() Get the
nodeIndex property of the satellite component.

tfamily \logicalzero nodeIndexIsSettable() Specify whether the nodeIndex property of the satellite
cOMPONENT is settable.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the satellite component
class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \logicalzero orbitingIsActive() Return whether the orbiting implementation of the satellite com-
ponent class is active.

tfamily \void output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

tfamily \void outputCount(<integer>integerPropertyCount↔, <integer>doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

tfamily \void outputNames(<integer>integerProperty↔, <char[\ast](::)>integerPropertyNames↔, <char[\ast](::)>integerPropertyComments↔, <double(:)>integerPropertyUnitsSI↔, <integer>doubleProperty↔, <char[\ast](::)>doublePropertyNames↔, <char[\ast](::)>doublePropertyComments↔, <double(:)>doublePropertyUnitsSI↔, <double> time→, <integer> instance→) Generate names of outputtable properties.

tfamily \textcolor{red}{\textless double(:)\textgreater} position() Get the position property of the satellite component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(::)\textgreater} positionAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the position property of the satellite component

tfamily \intzero positionCount() Compute the count of evolvable quantities in the position property of the SatelliteOrbiting component.

tfamily \textcolor{red}{\textless double(:)\textgreater} positionIsGettable() Get the position prop-
erty of the satellite component.

tfamily \logicalzero positionIsSettable() Specify whether the position property of the satellite comp-
ONENT is settable.

tfamily \void positionRate(<double(:)> value) Cumulate to the rate of the position property of the SatelliteOrbiting component.

tfamily \void positionScale(<double(:)> value) Set the scale of the position property of the SatelliteOrbiting component.

tfamily \void positionSet(<double(:)> value) Set the position property of the satellite component.

tfamily \doublezero potential(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <logical>[haloLoaded]→) Compute the gravitational potential.
tfamily \logicalzero presetIsActive() Return whether the preset implementation of the satellite component class is active.

tfamily \void readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezero rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the satellite component class is active.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the surface density.

tfamily \textcolor{red}{\textless double\textgreater} tidalHeatingNormalized() Get the tidalHeatingNormalized property of the satellite component.

tfamily \textcolor{red}{\textless type(varying\_string)(:)\textgreater} tidalHeatingNormalizedAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily \intzero tidalHeatingNormalizedCount() Compute the count of evolvable quantities in the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily \textcolor{red}{\textless double\textgreater} tidalHeatingNormalizedIsGettable() Get the tidalHeatingNormalized property of the satellite component.

tfamily \logicalzero tidalHeatingNormalizedIsSettable() Specify whether the tidalHeatingNormalized property of the satellite component is settable.

tfamily \void tidalHeatingNormalizedRate(<double> value) Cumulate to the rate of the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily \void tidalHeatingNormalizedScale(<double> value) Set the scale of the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily \void tidalHeatingNormalizedSet(<double> value) Set the tidalHeatingNormalized property of the satellite component.

tfamily \textcolor{red}{\textless type(tensorRank2Dimension3Symmetric):\textgreater} tidalTensorPathIntegrated() Get the tidalTensorPathIntegrated property of the satellite component.
tfamily \textcolor{red}\{	extless type(varying\_string)()\textgreater\} \textgreater tidalTensorPathIntegratedAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→)

Return a list of implementations that provide the given list of attributes for the tidalTensorPathIntegrated property of the satellite component.

tfamily \intzero tidalTensorPathIntegratedCount() Compute the count of evolvable quantities in the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily \textcolor{red}\{\textless type(tensorRank2Dimension3Symmetric)()\textgreater\} tidalTensorPathIntegratedIsGettable() Get the tidalTensorPathIntegrated property of the satellite component.

tfamily \logicalzero tidalTensorPathIntegratedIsSettable() Specify whether the tidalTensorPathIntegrated property of the satellite component is settable.

tfamily \void tidalTensorPathIntegratedRate(<type(tensorRank2Dimension3Symmetric)> value) Cummulate to the rate of the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily \void tidalTensorPathIntegratedScale(<type(tensorRank2Dimension3Symmetric)> value) Set the scale of the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily \void tidalTensorPathIntegratedSet(<type(tensorRank2Dimension3Symmetric)> value) Set the tidalTensorPathIntegrated property of the satellite component.

tfamily \textcolor{red}\{\textless double\textgreater\} timeOfMerging() Get the timeOfMerging property of the satellite component.

tfamily \textcolor{red}\{\textless type(varying\_string)()\textgreater\} timeOfMergingAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→)

Return a list of implementations that provide the given list of attributes for the timeOfMerging property of the satellite component.

tfamily \textcolor{red}\{\textless double\textgreater\} timeOfMergingIsGettable() Get the timeOfMerging property of the satellite component.

tfamily \logicalzero timeOfMergingIsSettable() Specify whether the timeOfMerging property of the satellite component is settable.

tfamily \void timeOfMergingSet(<double> value) Set the timeOfMerging property of the satellite component.

tfamily \textcolor{red}\{\textless type(varying\_string)()\textgreater\} type() Return the type of this object.

tfamily \textcolor{red}\{\textless double()\textgreater\} velocity() Get the velocity property of the satellite component.

tfamily \textcolor{red}\{\textless type(varying\_string)()\textgreater\} velocityAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→)

Return a list of implementations that provide the given list of attributes for the velocity property of the satellite component.

tfamily \intzero velocityCount() Compute the count of evolvable quantities in the velocity property of the SatelliteOrbiting component.

tfamily \textcolor{red}\{\textless double()\textgreater\} velocityIsGettable() Get the velocity property of the satellite component.
tfamily \textcolor{red}{\textless type(keplerOrbit)\textgreater} virialOrbit() Get the virialOrbit property of the satellite component.

tfamily \textcolor{red}{\textless double\textgreater} boundMass() Get the boundMass property of the satellite component.

tfamily \textcolor{red}{\textless type(keplerOrbit)\textgreater} virialOrbitAttributeMatch(\textcolor{red}{\textless logical\textgreater}[requireGettable] \rightarrow, \textcolor{red}{\textless logical\textgreater}[requireSettable] \rightarrow, \textcolor{red}{\textless logical\textgreater}[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the virialOrbit property of the satellite component.

tfamily \void virialOrbitFunction(<function()> deferredFunction) Set the function to be used for the get method of the virialOrbit property of the SatelliteStandard component.

tfamily \logicalzero virialOrbitIsAttached() Return whether the get method of the virialOrbit property of the SatelliteStandard component has been attached to a function.

tfamily \void virialOrbitSet(<type(keplerOrbit)> value) Set the virialOrbit property of the satellite component.

tfamily \void virialOrbitSetFunction(<function()> deferredFunction) Set the function to be used for the set method of the virialOrbit property of the SatelliteStandard component.

tfamily \logicalzero virialOrbitSetIsAttached() Return whether the set method of the virialOrbit property of the SatelliteStandard component has been attached to a function.

tfamily \void virialOrbitSetValue(<type(keplerOrbit)> value) Set the virialOrbit property of the satellite component.

tfamily \textcolor{red}{\textless double\textgreater} virialOrbitValue() Get the virialOrbit property of the satellite component.

tfamily nodeComponentSatelliteVerySimple

tfamily \logicalzero velocityIsSettable() Specify whether the velocity property of the satellite component is settable.

tfamily \void velocityRate(<double()> value) Cumulate to the rate of the velocity property of the SatelliteOrbiting component.

tfamily \void velocityScale(<double()> value) Set the scale of the velocity property of the SatelliteOrbiting component.

tfamily \void velocitySet(<double()> value) Set the velocity property of the satellite component.

tfamily \logicalzero verySimpleIsActive() Return whether the verySimple implementation of the satellite component class is active.

tfamily \void velocityRate(<double()> value) Cumulate to the rate of the velocity property of the SatelliteOrbiting component.

tfamily \void velocityScale(<double()> value) Set the scale of the velocity property of the SatelliteOrbiting component.

tfamily \void velocitySet(<double()> value) Set the velocity property of the satellite component.

tfamily \logicalzero verySimpleIsActive() Return whether the verySimple implementation of the satellite component class is active.

644
16.5. Objects

tfamily \textcolor{red}{\textless} intzero boundMassCount() Compute the count of evolvable quantities in the boundMass property of the SatelliteStandard component.

tfamily \textcolor{red}{\textless} type(history)\textgreater boundMassHistory() Get the boundMassHistory property of the satellite component.

tfamily \textcolor{red}{\textless} type(varying\_string)(:)\textgreater boundMassHistoryAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the boundMassHistory property of the satellite component.

tfamily \textcolor{red}{\textless} type(history)\textgreater boundMassHistoryIsGettable() Get the boundMassHistory property of the satellite component.

tfamily \textcolor{red}{\textless} type(history)\textgreater boundMassHistoryIsSettable() Specify whether the boundMassHistory property of the satellite component is settable.

tfamily \void boundMassHistorySet(<type(history)> value) Set the boundMassHistory property of the satellite component.

tfamily \void destruction boundMassScale(<double> value) Set the scale of the boundMass property of the SatelliteStandard component.

tfamily \void boundMassSet(<double> value) Set the boundMass property of the satellite component.

tfamily \void builder(<type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily \doublezero density(<double(3)> positionSpherical→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.
tfamily\[\text{double}\]enclosedMass(\text{<double>}radius\rightarrow,\text{<componentType>}[\text{componentType}]\rightarrow,\text{<massType>}[\text{massType}]\rightarrow,\text{<weightBy>}[\text{weightBy}]\rightarrow,\text{<integer>}[\text{weightIndex}]\rightarrow,\text{<logical>}[\text{haloLoaded}]\rightarrow)
Compute the mass enclosed within a radius.

tfamily\{\text{red}\}{*\text{type}(\text{treeNode})*}\text{host()}\text{Return a pointer to the host treeNode object.}

tfamily\text{void hotHaloCoolingAbundancesRate(\text{<type}(\text{abundances})})\text{value})\text{Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.}

tfamily\text{void hotHaloCoolingAngularMomentumRate(\text{<double>})\text{value})\text{Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.}

tfamily\text{void hotHaloCoolingMassRate(\text{<double>})\text{value})\text{Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.}

tfamily\text{void initialize() Initialize the object.}

tfamily\{\text{red}\}{\text{double}}mergeTime()\text{Get the mergeTime property of the satellite component.}

tfamily\{\text{red}\}{\text{integer}}nodeIndex()\text{Get the nodeIndex property of the satellite component.}

tfamily\{\text{red}\}{\text{integer}}mergeTimeCount()\text{Compute the count of evolvable quantities in the mergeTime property of the SatelliteVerySimple component.}

tfamily\{\text{red}\}{\text{double}}mergeTimeIsGettable()\text{Get the mergeTime property of the satellite component.}

tfamily\{\text{red}\}{\text{double}}mergeTimeSet(<\text{double}>value)\text{Set the mergeTime property of the SatelliteVerySimple component.}

tfamily\{\text{red}\}{\text{string}}nodeIndexAttributeMatch(\text{<logical>}[\text{requireGettable}]\rightarrow,\text{<logical>}[\text{requireSettable}]\rightarrow,\text{<logical>}[\text{requireEvolvable}]\rightarrow)\text{Return a list of implementations that provide the given list off attributes for the mergeTime property of the satellite component}.

tfamily\{\text{red}\}{\text{logical}}mergeTimeIsSettable()\text{Specify whether the mergeTime property of the satellite component is settable.}

tfamily\text{void mergeTimeRate(\text{<double>})\text{value})\text{Cumulate to the rate of the mergeTime property of the SatelliteVerySimple component.}

tfamily\text{void mergeTimeScale(\text{<double>})\text{value})\text{Set the scale of the mergeTime property of the SatelliteVerySimple component.}

tfamily\text{void nameFromIndex(\text{<integer>}count\rightarrow,\text{<varying_string>}name\leftarrow)\text{Return the name of a property given is index.}

tfamily\{\text{red}\}{\text{integer}}nodeIndex()\text{Get the nodeIndex property of the satellite component.}

tfamily\{\text{red}\}{\text{integer}}nodeIndexAttributeMatch(\text{<logical>}[\text{requireGettable}]\rightarrow,\text{<logical>}[\text{requireSettable}]\rightarrow,\text{<logical>}[\text{requireEvolvable}]\rightarrow)\text{Return a list of implementations that provide the given list off attributes for the nodeIndex property of the satellite component}.

tfamily\{\text{red}\}{\text{integer}}nodeIndexHistory()\text{Get the nodeIndexHistory property of the satellite component.}
16.5. Objects

tfamily \textcolor{red}{\textless \textsf{type(varying\_string)}():\textgreater} nodeIndexHistoryAttributeMatch(<\textsf{logical}:[requireGettable]\rightarrow,<\textsf{logical}:[requireSettable]\rightarrow,<\textsf{logical}:[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the nodeIndexHistory property of the satellite component.

tfamily \textcolor{red}{\textless \textsf{type(longIntegerHistory)}():\textgreater} nodeIndexHistoryIsGettable() Get the nodeIndexHistory property of the satellite component.

tfamily \textcolor{red}{\textless \textsf{type(longIntegerHistory)}():\textgreater} nodeIndexHistoryIsSettable() Specify whether the nodeIndexHistory property of the satellite component is settable.

tfamily \textcolor{red}{\textless \textsf{type(longIntegerHistory)}():\textgreater} nodeIndexHistorySet(<\textsf{type(longIntegerHistory)}> value) Set the nodeIndexHistory property of the satellite component.

tfamily \textcolor{red}{\textless \textsf{type(longIntegerHistory)}():\textgreater} nodeIndexIsGettable() Get the nodeIndex property of the satellite component.

tfamily \textcolor{red}{\textless \textsf{type(longIntegerHistory)}():\textgreater} nodeIndexIsSettable() Specify whether the nodeIndex property of the satellite component is settable.

tfamily \textcolor{red}{\textless \textsf{type(longIntegerHistory)}():\textgreater} nullIsActive() Return whether the null implementation of the satellite component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \textcolor{red}{\textless \textsf{type(longIntegerHistory)}():\textgreater} orbitingIsActive() Return whether the orbiting implementation of the satellite component class is active.

tfamily \void output(<\textsf{integer}> integerProperty\leftrightarrow,<\textsf{integer}> integerBufferCount\leftrightarrow,<\textsf{integer}(;;)> integerBuffer\leftrightarrow,<\textsf{integer}> doubleProperty\leftrightarrow,<\textsf{integer}> doubleBufferCount\leftrightarrow,<\textsf{double}(;;)> doubleBuffer\leftrightarrow,<\textsf{double}> time\rightarrow,<\textsf{integer}> instance\rightarrow) Generate values of outputtable properties.

tfamily \void outputCount(<\textsf{integer}> integerPropertyCount\leftrightarrow,<\textsf{integer}> doublePropertyCount\leftrightarrow,<\textsf{double}> time\rightarrow,<\textsf{integer}> instance\rightarrow) Compute a count of outputtable properties.

tfamily \void outputNames(<\textsf{integer}> integerProperty\leftrightarrow,<\textsf{char}[*]> integerPropertyNames\leftrightarrow,<\textsf{char}[*]> integerPropertyComments\leftrightarrow,<\textsf{double}()> integerPropertyUnitsSI\leftrightarrow,<\textsf{integer}> doubleProperty\leftrightarrow,<\textsf{char}[*]> doublePropertyNames\leftrightarrow,<\textsf{char}[*]> doublePropertyComments\leftrightarrow,<\textsf{double}()> doublePropertyUnitsSI\leftrightarrow,<\textsf{double}> time\rightarrow,<\textsf{integer}> instance\rightarrow) Generate names of outputtable properties.

tfamily \textcolor{red}{\textless \textsf{double}(;;):\textgreater} position() Get the position property of the satellite component.

tfamily \textcolor{red}{\textless \textsf{type(varying\_string)(;)::textgreater}} positionAttributeMatch(<\textsf{logical}:[requireGettable]\rightarrow,<\textsf{logical}:[requireSettable]\rightarrow,<\textsf{logical}:[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the position property of the satellite component.

tfamily \int zero positionCount() Compute the count of evolvable quantities in the position property of the SatelliteOrbiting component.

tfamily \textcolor{red}{\textless \textsf{double}(;;):\textgreater} positionIsGettable() Get the position property of the satellite component.
16. Coding Galacticus

tfamily \logicalzero positionIsSettable() Specify whether the position property of the satellite component is settable.

tfamily \void positionRate(<double():> value) Cumulate to the rate of the position property of the SatelliteOrbiting component.

tfamily \void positionScale(<double():> value) Set the scale of the position property of the SatelliteOrbiting component.

tfamily \void positionSet(<double():> value) Set the position property of the satellite component.

tfamily \double zero potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

tfamily \logicalzero presetIsActive() Return whether the preset implementation of the satellite component class is active.

tfamily \void readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \double zero rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

tfamily \double zero rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double():> array →) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double():> array →) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double():> array →) Serialize the evolvable quantities to an array.

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the satellite component class is active.

tfamily \double zero surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the surface density.

tfamily \textcolor{red}{\textless double\textgreater} tidalHeatingNormalized() Get the tidalHeatingNormalized property of the satellite component.

tfamily \textcolor{red}{\textless varying\_string\rangle::\textgreater} tidalHeatingNormalizedAttributeMatch(<logical> [requireGettable] →, <logical> [requireSettable] →, <logical> [requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the tidalHeatingNormalized property of the satellite component.

tfamily \intzero tidalHeatingNormalizedCount() Compute the count of evolvable quantities in the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily \textcolor{red}{\textless double\textgreater} tidalHeatingNormalizedIsGettable() Get the tidalHeatingNormalized property of the satellite component.

tfamily \logicalzero tidalHeatingNormalizedIsSettable() Specify whether the tidalHeatingNormalized property of the satellite component is settable.
16.5. Objects

tfamily \void tidalHeatingNormalizedRate(<double> value) Cumulate to the rate of the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily \void tidalHeatingNormalizedScale(<double> value) Set the scale of the tidalHeatingNormalized property of the SatelliteOrbiting component.

tfamily \void tidalHeatingNormalizedSet(<double> value) Set the tidalHeatingNormalized property of the satellite component.

tfamily \textcolor{red}{\textless type(tensorRank2Dimension3Symmetric)\textgreater} tidalTensorPathIntegrated() Get the tidalTensorPathIntegrated property of the satellite component.

tfamily \textcolor{red}{\textless type(varying_string)(:)\textgreater} tidalTensorPathIntegratedAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the tidalTensorPathIntegrated property of the satellite component.

tfamily \intzero tidalTensorPathIntegratedCount() Compute the count of evolvable quantities in the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily \textcolor{red}{\textless type(tensorRank2Dimension3Symmetric)\textgreater} tidalTensorPathIntegratedIsGettable() Get the tidalTensorPathIntegrated property of the satellite component.

tfamily \logicalzero tidalTensorPathIntegratedIsSettable() Specify whether the tidalTensorPathIntegrated property of the satellite component is settable.

tfamily \void tidalTensorPathIntegratedRate(<type(tensorRank2Dimension3Symmetric)> value) Cumulate to the rate of the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily \void tidalTensorPathIntegratedScale(<type(tensorRank2Dimension3Symmetric)> value) Set the scale of the tidalTensorPathIntegrated property of the SatelliteOrbiting component.

tfamily \void tidalTensorPathIntegratedSet(<type(tensorRank2Dimension3Symmetric)> value) Set the tidalTensorPathIntegrated property of the satellite component.

tfamily \textcolor{red}{\textless double\textgreater} timeOfMerging() Get the timeOfMerging property of the satellite component.

tfamily \textcolor{red}{\textless type(varying_string)(:)\textgreater} timeOfMergingAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the timeOfMerging property of the satellite component.

tfamily \textcolor{red}{\textless double\textgreater} timeOfMergingIsGettable() Get the timeOfMerging property of the satellite component.

tfamily \logicalzero timeOfMergingIsSettable() Specify whether the timeOfMerging property of the satellite component is settable.

tfamily \void timeOfMergingSet(<double> value) Set the timeOfMerging property of the satellite component.

tfamily \textcolor{red}{\textless type(varying_string)\textgreater} type() Return the type of this object.
tfamily \textcolor{red}{\textless double(:)\textgreater} velocity() Get the velocity property of the satellite component.

tfamily \textcolor{red}{\textless type(varying\_string):\textgreater} velocityAttributeMatch(\textcolor{red}{\textless logical:\textgreater[requireGettable]} →, \textcolor{red}{\textless logical:\textgreater[requireSettable]} →, \textcolor{red}{\textless logical:\textgreater[requireEvolvable]} →) Return a list of implementations that provide the given list off attributes for the velocity property of the satellite component.

tfamily \intzero velocityCount() Compute the count of evolvable quantities in the velocity property of the SatelliteOrbiting component.

tfamily \textcolor{red}{\textless double(:)\textgreater} velocityIsGettable() Get the velocity property of the satellite component.

tfamily \logicalzero velocityIsSettable() Specify whether the velocity property of the satellite component is settable.

tfamily \void velocityRate(<double(:)> value) Cumulate to the rate of the velocity property of the SatelliteOrbiting component.

tfamily \void velocityScale(<double(:)> value) Set the scale of the velocity property of the SatelliteOrbiting component.

tfamily \void velocitySet(<double(:)> value) Set the velocity property of the satellite component.

tfamily \logicalzero verySimpleIsActive() Return whether the verySimple implementation of the satellite component class is active.

tfamily \textcolor{red}{\textless type(keplerOrbit):\textgreater} virialOrbit() Get the virialOrbit property of the satellite component.

tfamily \textcolor{red}{\textless type(varying\_string):\textgreater} virialOrbitAttributeMatch(\textcolor{red}{\textless logical:\textgreater[requireGettable]} →, \textcolor{red}{\textless logical:\textgreater[requireSettable]} →, \textcolor{red}{\textless logical:\textgreater[requireEvolvable]} →) Return a list of implementations that provide the given list off attributes for the virialOrbit property of the satellite component.

tfamily \void virialOrbitSet(<type(keplerOrbit)> value) Set the virialOrbit property of the satellite component.

tfamily \logicalzero virialOrbitIsSettable() Specify whether the virialOrbit property of the satellite component is settable.

tfamily \void virialOrbitSet(<type(keplerOrbit)> value) Set the virialOrbit property of the satellite component.

tfamily nodeComponentSpheroid

tfamily \textcolor{red}{\textless type(keplerOrbit):\textgreater} abundancesGas() Get the abundancesGas property of the spheroid component.

tfamily \textcolor{red}{\textless type(varying\_string):\textgreater} abundancesGasAttributeMatch(\textcolor{red}{\textless logical:\textgreater[requireGettable]} →, \textcolor{red}{\textless logical:\textgreater[requireSettable]} →, \textcolor{red}{\textless logical:\textgreater[requireEvolvable]} →) Return a list of implementations that provide the given list off attributes for the abundancesGas property of the spheroid component.

tfamily \intzero abundancesGasCount() Compute the count of evolvable quantities in the abundancesGas property of the SpheroidStandard component.
16.5. Objects

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesGasIsGettable() Get the
abundancesGas property of the spheroid component.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} abundancesGasIsSettable() Specify whether the abundancesGas property of the
spheroid component is settable.

tfamily void abundancesGasRate(<type(abundances)> value) Cumulate to the rate of the abundancesGas
property of the SpheroidStandard component.

tfamily void abundancesGasScale(<type(abundances)> value) Set the scale of the abundancesGas
property of the SpheroidStandard component.

tfamily void abundancesGasSet(<type(abundances)> value) Set the abundancesGas property of the
spheroid component.

abundancesGasRate(<type(abundances)>value)

abundancesGasScale(<type(abundances)> value)

abundancesGasSet(<type(abundances)> value)

abundancesGasIsGettable()

abundancesGasIsSettable()

abundancesGasRate(<type(abundances)> value)

abundancesGasScale(<type(abundances)> value)

abundancesGasSet(<type(abundances)> value)

abundancesGasIsGettable()

abundancesGasIsSettable()

abundancesGasRate(<type(abundances)> value)

abundancesGasScale(<type(abundances)> value)

abundancesGasSet(<type(abundances)> value)

abundancesGasIsGettable()

abundancesGasIsSettable()

abundancesGasRate(<type(abundances)> value)

abundancesGasScale(<type(abundances)> value)

abundancesGasSet(<type(abundances)> value)

abundancesGasIsGettable()

abundancesGasIsSettable()

abundancesGasRate(<type(abundances)> value)

abundancesGasScale(<type(abundances)> value)

abundancesGasSet(<type(abundances)> value)

abundancesGasIsGettable()

abundances_gas_is_settable()

abundances_gas_rate(<type(abundances)> value)

abundances_gas_scale(<type(abundances)> value)

abundances_gas_set(<type(abundances)> value)

abundances_gas_is_gettable()

abundances_gas_is_settable()
tfamily logical zero angularMomentumIsSettable() Specify whether the \textit{angularMomentum} property of the \textit{spheroid} component is settable.

tfamily void angularMomentumRate(\textcolor{red}{\textless} double \textcolor{red}{\textgreater} value) Cumulate to the rate of the \textit{angularMomentum} property of the \textit{SpheroidStandard} component.

tfamily void angularMomentumScale(\textcolor{red}{\textless} double \textcolor{red}{\textgreater} value) Set the scale of the \textit{angularMomentum} property of the \textit{SpheroidStandard} component.

tfamily void angularMomentumSet(\textcolor{red}{\textless} double \textcolor{red}{\textgreater} value) Set the \textit{angularMomentum} property of the \textit{spheroid} component.

tfamily void builder(\textcolor{red}{\textless} \textit{type(node)} \textcolor{red}{\textgreater} componentDefinition \textcolor{red}{\rightarrow}) Build a \textit{nodeComponent} from a supplied XML definition.

tfamily double zero density(\textcolor{red}{\textless} double(3) \textcolor{red}{\textgreater} positionSpherical \textcolor{red}{\rightarrow}, \textcolor{red}{\textless} \textit{componentType} \textcolor{red}{\textgreater} [\textcolor{red}{\textgreater} componentType \textcolor{red}{\rightarrow}], \textcolor{red}{\textless} \textit{massType} \textcolor{red}{\textgreater} [\textcolor{red}{\textgreater} massType \textcolor{red}{\rightarrow}], \textcolor{red}{\textless} \textit{weightBy} \textcolor{red}{\textgreater} [\textcolor{red}{\textgreater} weightBy \textcolor{red}{\rightarrow}], \textcolor{red}{\textless} \textit{integer} \textcolor{red}{\textgreater} [\textcolor{red}{\textgreater} weightIndex \textcolor{red}{\rightarrow}], \textcolor{red}{\textless} \textit{logical} \textcolor{red}{\textgreater} [\textcolor{red}{\textgreater} haloLoaded \textcolor{red}{\rightarrow}]) Compute the density.

tfamily void deserializeRates(\textcolor{red}{\textless} double(\textcolor{red}{\textgreater}) array \textcolor{red}{\leftarrow}) Deserialize the evolvable rates from an array.

tfamily void deserializeScales(\textcolor{red}{\textless} double(\textcolor{red}{\textgreater}) array \textcolor{red}{\leftarrow}) Deserialize the evolvable scales from an array.

tfamily void deserializeValues(\textcolor{red}{\textless} double(\textcolor{red}{\textgreater}) array \textcolor{red}{\leftarrow}) Deserialize the evolvable quantities from an array.

tfamily void destroy() Destroy the object.

tfamily void dump() Generate an ASCII dump of all properties.

tfamily void dumpRaw(\textcolor{red}{\textless} \textit{integer} \textcolor{red}{\textgreater} fileHandle \textcolor{red}{\rightarrow}) Generate a binary dump of all properties.

tfamily void dumpXML() Generate an XML dump of all properties.

tfamily double zero enclosedMass(\textcolor{red}{\textless} double \textcolor{red}{\textgreater} radius \textcolor{red}{\rightarrow}, \textcolor{red}{\textless} \textit{componentType} \textcolor{red}{\textgreater} [\textcolor{red}{\textgreater} componentType \textcolor{red}{\rightarrow}], \textcolor{red}{\textless} \textit{massType} \textcolor{red}{\textgreater} [\textcolor{red}{\textgreater} massType \textcolor{red}{\rightarrow}], \textcolor{red}{\textless} \textit{weightBy} \textcolor{red}{\textgreater} [\textcolor{red}{\textgreater} weightBy \textcolor{red}{\rightarrow}], \textcolor{red}{\textless} \textit{integer} \textcolor{red}{\textgreater} [\textcolor{red}{\textgreater} weightIndex \textcolor{red}{\rightarrow}], \textcolor{red}{\textless} \textit{logical} \textcolor{red}{\textgreater} [\textcolor{red}{\textgreater} haloLoaded \textcolor{red}{\rightarrow}]) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless} \textit{double(\textcolor{red}{\textgreater}) \textcolor{red}{\textgreater}} energyGasInput() Get the \textit{energyGasInput} property of the \textit{spheroid} component.

tfamily \textcolor{red}{\textless} \textit{double(\textcolor{red}{\textgreater}) \textcolor{red}{\textgreater}} energyGasInputAttributeMatch(\textcolor{red}{\textless} \textit{logical(\textcolor{red}{\textgreater}) \textcolor{red}{\textgreater}} requireGettable \textcolor{red}{\rightarrow}, \textcolor{red}{\textless} \textit{logical(\textcolor{red}{\textgreater}) \textcolor{red}{\textgreater}} requireSettable \textcolor{red}{\rightarrow}, \textcolor{red}{\textless} \textit{logical(\textcolor{red}{\textgreater}) \textcolor{red}{\textgreater}} requireEvolvable \textcolor{red}{\rightarrow}) Return a list of implementations that provide the given list off attributes for the \textit{energyGasInput} property of the \textit{spheroid} component.

tfamily int zero energyGasInputCount() Compute the count of evolvable quantities in the \textit{energyGasInput} property of the \textit{SpheroidStandard} component.

tfamily \textcolor{red}{\textless} \textit{double(\textcolor{red}{\textgreater}) \textcolor{red}{\textgreater}} energyGasInputIsGettable() Get the \textit{energyGasInput} property of the \textit{spheroid} component.

tfamily logical zero energyGasInputIsSettable() Specify whether the \textit{energyGasInput} property of the \textit{spheroid} component is settable.

tfamily void energyGasInputRate(\textcolor{red}{\textless} double \textcolor{red}{\textgreater} value) Cumulate to the rate of the \textit{energyGasInput} property of the \textit{SpheroidStandard} component.
16.5. Objects

- halfMassRadius() Get the halfMassRadius property of the spheroid component.

- halfMassRadiusAttributeMatch() Return a list of implementations that provide the given list of attributes for the halfMassRadius property of the spheroid component.

- halfMassRadiusIsGettable() Get the halfMassRadius property of the spheroid component.

- halfMassRadiusIsSettable() Specify whether the halfMassRadius property of the spheroid component is settable.

- host() Return a pointer to the host TreeNode object.

- hotHaloCoolingAbundancesRate() Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

- hotHaloCoolingAngularMomentumRate() Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

- hotHaloCoolingMassRate() Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

- initialize() Initialize the object.

- isInitialized() Get the isInitialized property of the spheroid component.

- isInitializedAttributeMatch() Return a list of implementations that provide the given list of attributes for the isInitialized property of the spheroid component.

- isInitializedIsGettable() Get the isInitialized property of the spheroid component.

- isInitializedIsSettable() Specify whether the isInitialized property of the spheroid component is settable.

- isInitializedSet() Set the isInitialized property of the spheroid component.

- luminositiesStellar() Get the luminositiesStellar property of the spheroid component.

- luminositiesStellarAttributeMatch() Return a list of implementations that provide the given list of attributes for the luminositiesStellar property of the spheroid component.

- luminositiesStellarCount() Compute the count of evolvable quantities in the luminositiesStellar property of the SpheroidStandard component.

- luminositiesStellarIsGettable() Get the luminositiesStellar property of the spheroid component.
tfamily \textcolor{red}{\textless \textgreater} luminositiesStellarIsSettable() Specify whether the luminositiesStellar property of the spheroid component is settable.

tfamily \void luminositiesStellarRate(\textcolor{type(stellarLuminosities)} value) Cumulate to the rate of the luminositiesStellar property of the SpheroidStandard component.

tfamily \void luminositiesStellarScale(\textcolor{type(stellarLuminosities)} value) Set the scale of the luminositiesStellar property of the SpheroidStandard component.

tfamily \void luminositiesStellarSet(\textcolor{type(stellarLuminosities)} value) Set the luminositiesStellar property of the spheroid component.

tfamily \textcolor{red}{\textless \textgreater} massGas() Get the massGas property of the spheroid component.

tfamily \textcolor{red}{\textless \textgreater} massGasAttributeMatch(\textcolor{logical\[requireGettable\]} \rightarrow, \textcolor{logical\[requireSettable\]} \rightarrow, \textcolor{logical\[requireEvolvable\]} \rightarrow) Return a list of implementations that provide the given list off attributes for the massGas property of the spheroid component

tfamily \intzero massGasCount() Compute the count of evolvable quantities in the massGas property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless \textgreater} massGasIsGettable() Get the massGas property of the spheroid component.

tfamily \logicalzero massGasIsSettable() Specify whether the massGas property of the spheroid component is settable.

tfamily \void massGasRate(\textcolor{double} value) Cumulate to the rate of the massGas property of the SpheroidStandard component.

tfamily \void massGasScale(\textcolor{double} value) Set the scale of the massGas property of the SpheroidStandard component.

tfamily \void massGasSet(\textcolor{double} value) Set the massGas property of the spheroid component.

tfamily \textcolor{red}{\textless \textgreater} massGasSink() Get the massGasSink property of the spheroid component.

tfamily \textcolor{red}{\textless \textgreater} massGasSinkAttributeMatch(\textcolor{logical\[requireGettable\]} \rightarrow, \textcolor{logical\[requireSettable\]} \rightarrow, \textcolor{logical\[requireEvolvable\]} \rightarrow) Return a list of implementations that provide the given list off attributes for the massGasSink property of the spheroid component

tfamily \intzero massGasSinkCount() Compute the count of evolvable quantities in the massGasSink property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless \textgreater} massGasSinkIsGettable() Get the massGasSink property of the spheroid component.

tfamily \logicalzero massGasSinkIsSettable() Specify whether the massGasSink property of the spheroid component is settable.

tfamily \void massGasSinkRate(\textcolor{double} value) Cumulate to the rate of the massGasSink property of the SpheroidStandard component.
16.5. Objects

tfamily \textcolor{red}{\textless double\textgreater} massStellar() Get the massStellar property of the spheroid component.

tfamily \textcolor{red}{\textless double\textgreater} massStellarAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the massStellar property of the spheroid component.

tfamily \intzero massStellarCount() Compute the count of evolvable quantities in the massStellar property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless double\textgreater} massStellarIsGettable() Get the massStellar property of the spheroid component.

tfamily \logicalzero massStellarIsSettable() Specify whether the massStellar property of the spheroid component is settable.

tfamily \void massStellarRate(<double> value) Cumulate to the rate of the massStellar property of the SpheroidStandard component.

tfamily \void massStellarScale(<double> value) Set the scale of the massStellar property of the SpheroidStandard component.

tfamily \void massStellarSet(<double> value) Set the massStellar property of the spheroid component.

tfamily \void nameFromIndex(<integer> count →, <varying_string> name ←) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the spheroid component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer> doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty ↔, <char[*](:)> integerPropertyNames ↔, <char[*](:)> integerPropertyComments ↔, <double(:)> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char[*](:)> doublePropertyNames ↔, <char[*](:)> doublePropertyComments ↔, <double(:)> doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

tfamily \doublezero potential(<double> radius →, <componentType>[componentType] →, <massType>[massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

tfamily \textcolor{red}{\textless textless double\textgreater} radius() Get the radius property of the spheroid component.
16. Coding Galacticus

tfamily \textcolor{red}{\textless type(varying_string):>}\textgreater radiusAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the radius property of the spheroid component.

tfamily \textcolor{red}{\textless double\textgreater} radiusIsGettable() Get the radius property of the spheroid component.

tfamily \textcolor{red}{\textless type(history):>}\textgreater\textless double\textgreater starFormationRate() Get the starFormationRate property of the spheroid component.

tfamily \textcolor{red}{\textless type(history):>\textgreater} starFormationHistory() Get the starFormationHistory property of the spheroid component.

tfamily \textcolor{red}{\textless type(history):>\textgreater} starFormationHistoryAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the starFormationHistory property of the spheroid component.

tfamily \intzero starFormationHistoryCount() Compute the count of evolvable quantities in the starFormationHistory property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless type(history):>\textgreater} starFormationHistoryIsGettable() Get the starFormationHistory property of the spheroid component.

tfamily \textcolor{red}{\textless type(history):>\textgreater} starFormationHistoryIsSettable() Specify whether the starFormationHistory property of the spheroid component is settable.

tfamily \textcolor{red}{\textless type(history):>\textgreater} starFormationHistoryRate(<type(history)> value) Cumulate to the rate of the starFormationHistory property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless type(history):>\textgreater} starFormationHistoryScale(<type(history)> value) Set the scale of the starFormationHistory property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless type(history):>\textgreater} starFormationHistorySet(<type(history)> value) Set the starFormationHistory property of the spheroid component.

tfamily \textcolor{red}{\textless double\textgreater} starFormationRate() Get the starFormationRate property of the spheroid component.
16.5. Objects

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} starFormationRateAttributeMatch(<logical> [requireGettable]\rightarrow, <logical> [requireSettable]\rightarrow, <logical> [requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the starFormationRate property of the spheroid component.

tfamily \textcolor{red}{\textless double\textgreater} starFormationRateIsGettable() Get the starFormationRate property of the spheroid component.

tfamily \textcolor{red}{\textless boolean\textgreater} starFormationRateIsSettable() Specify whether the starFormationRate property of the spheroid component is settable.

tfamily \textcolor{red}{\textless type(history)\textgreater} stellarPropertiesHistory() Get the stellarPropertiesHistory property of the spheroid component.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} stellarPropertiesHistoryAttributeMatch(<logical> [requireGettable]\rightarrow, <logical> [requireSettable]\rightarrow, <logical> [requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the stellarPropertiesHistory property of the spheroid component.

tfamily \int zero stellarPropertiesHistoryCount() Compute the count of evolvable quantities in the stellarPropertiesHistory property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless type(history)\textgreater} stellarPropertiesHistoryIsGettable() Get the stellarPropertiesHistory property of the spheroid component.

tfamily \textcolor{red}{\textless boolean\textgreater} stellarPropertiesHistoryIsSettable() Specify whether the stellarPropertiesHistory property of the spheroid component is settable.

tfamily \textcolor{red}{\textless type(history)>\textless double\textgreater} stellarPropertiesHistoryRate(<type(history)> value) Cumulate to the rate of the stellarPropertiesHistory property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless type(history)>\textless double\textgreater} stellarPropertiesHistoryScale(<type(history)> value) Set the scale of the stellarPropertiesHistory property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless type(history)>\textless double\textgreater} stellarPropertiesHistorySet(<type(history)> value) Set the stellarPropertiesHistory property of the spheroid component.

tfamily \textcolor{red}{\textless double\textgreater} surfaceDensity(<double(3)> positionCylindrical\rightarrow, <componentType> [componentType]\rightarrow, <massType> [massType]\rightarrow, <weightBy> [weightBy]\rightarrow, <integer> [weightIndex]\rightarrow, <logical> [haloLoaded]\rightarrow) Compute the surface density.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

tfamily \textcolor{red}{\textless double\textgreater} velocity() Get the velocity property of the spheroid component.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} velocityAttributeMatch(<logical> [requireGettable]\rightarrow, <logical> [requireSettable]\rightarrow, <logical> [requireEvolvable]\rightarrow) Return a list of implementations that provide the given list of attributes for the velocity property of the spheroid component.

tfamily \textcolor{red}{\textless double\textgreater} velocityIsGettable() Get the velocity property of the spheroid component.

tfamily \textcolor{red}{\textless boolean\textgreater} velocityIsSettable() Specify whether the velocity property of the spheroid component is settable.

tfamily \textcolor{red}{\textless double\textgreater} velocitySet(<double> value) Set the velocity property of the spheroid component.
tfamily nodeComponentSpheroidNull

- abundancesGas() Get the abundancesGas property of the spheroid component.
- abundancesGasAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the abundancesGas property of the spheroid component.
- abundancesGasCount() Compute the count of evolvable quantities in the abundancesGas property of the SpheroidStandard component.
- abundancesGasIsGettable() Get the abundancesGas property of the spheroid component.
- abundancesGasIsSettable() Specify whether the abundancesGas property of the spheroid component is settable.
- abundancesGasRate(<type(abundances)> value) Cumulate to the rate of the abundancesGas property of the SpheroidStandard component.
- abundancesGasScale(<type(abundances)> value) Set the scale of the abundancesGas property of the SpheroidStandard component.
- abundancesGasSet(<type(abundances)> value) Set the abundancesGas property of the spheroid component.

- abundancesStellar() Get the abundancesStellar property of the spheroid component.
- abundancesStellarAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the abundancesStellar property of the spheroid component.
- abundancesStellarCount() Compute the count of evolvable quantities in the abundancesStellar property of the SpheroidStandard component.
- abundancesStellarIsGettable() Get the abundancesStellar property of the spheroid component.
- abundancesStellarIsSettable() Specify whether the abundancesStellar property of the spheroid component is settable.
- abundancesStellarRate(<type(abundances)> value) Cumulate to the rate of the abundancesStellar property of the SpheroidStandard component.
- abundancesStellarScale(<type(abundances)> value) Set the scale of the abundancesStellar property of the SpheroidStandard component.
- abundancesStellarSet(<type(abundances)> value) Set the abundancesStellar property of the spheroid component.

- angularMomentum() Get the angularMomentum property of the spheroid component.
16.5. Objects

tfamily \textcolor{red}{\textlesstype(varying \\ _string):}{\textgreater} angularMomentumAttributeMatch(<logical>[requireGettable] {\rightarrow}, <logical>[requireSettable] {\rightarrow}, <logical>[requireEvolvable] {\rightarrow})

Return a list of implementations that provide the given list of attributes for the angularMomentum property of the spheroid component.

tfamily \textcolor{red}{\textless double \textgreater} angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the SpheroidStandard component.

tfamily \textcolor{red}{\textlesstype(varying \_string):}{\textgreater} angularMomentumIsGettable() Get the angularMomentum property of the spheroid component.

tfamily \textcolor{red}{\textlesstype(varying \_string):}{\textgreater} angularMomentumIsSettable() Specify whether the angularMomentum property of the spheroid component is settable.

tfamily \textcolor{red}{\textlesstype(varying \_string):}{\textgreater} angularMomentumSet(<double> value) Set the angularMomentum property of the spheroid component.

tfamily \textcolor{red}{\textlesstype(varying \_string):}{\textgreater} angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the SpheroidStandard component.

tfamily \textcolor{red}{\textlesstype(varying \_string):}{\textgreater} angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the SpheroidStandard component.

tfamily \textcolor{red}{\textlesstype(varying \_string):}{\textgreater} angularMomentumSet(<double> value) Set the angularMomentum property of the spheroid component.

tfamily \textcolor{red}{\textlesstype(varying \_string):}{\textgreater} energyGasInputAttributeMatch(<logical>[requireGettable] {\rightarrow}, <logical>[requireSettable] {\rightarrow}, <logical>[requireEvolvable] {\rightarrow})

Return a list of implementations that provide the given list of attributes for the energyGasInput property of the spheroid component.

tfamily \textcolor{red}{\textlesstype(varying \_string):}{\textgreater} energyGasInputCount() Compute the count of evolvable quantities in the energyGasInput property of the SpheroidStandard component.
tfamily \textcolor{red}{\textless double\textgreater} energyGasInputIsGettable() Get the energyGasInput property of the spheroid component.

tfamily \textcolor{red}{\textless logical\textgreater} energyGasInputIsSettable() Specify whether the energyGasInput property of the spheroid component is settable.

tfamily \textcolor{red}{\textless double\textgreater} energyGasInputRate(<double> value) Cumulate to the rate of the energyGasInput property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless double\textgreater} halfMassRadius() Get the halfMassRadius property of the spheroid component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)}\textgreater halfMassRadiusAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list off attributes for the halfMassRadius property of the spheroid component.

tfamily \textcolor{red}{\textless double\textgreater} halfMassRadiusIsGettable() Get the halfMassRadius property of the spheroid component.

tfamily \textcolor{red}{\textless logical\textgreater} halfMassRadiusIsSettable() Specify whether the halfMassRadius property of the spheroid component is settable.

tfamily \textcolor{red}{\textless type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \textcolor{red}{\textless type(abundances)\textgreater} hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \textcolor{red}{\textless double\textgreater} hotHaloCoolingMassRate(<double>value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \textcolor{red}{\textless logical\textgreater} isInitialized() Get the isInitialized property of the spheroid component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)}\textgreater isInitializedAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list off attributes for the isInitialized property of the spheroid component.

tfamily \textcolor{red}{\textless logical\textgreater} isInitializedIsGettable() Get the isInitialized property of the spheroid component.

tfamily \textcolor{red}{\textless logical\textgreater} isInitializedIsSettable() Specify whether the isInitialized property of the spheroid component is settable.

tfamily \textcolor{red}{\textless logical\textgreater} isInitializedSet(<logical> value) Set the isInitialized property of the spheroid component.

tfamily \textcolor{red}{\textless type(stellarLuminosities)\textgreater} luminositiesStellar() Get the luminositiesStellar property of the spheroid component.
16.5. Objects

tfamily \textcolor{red}{{\textless type(varying\_string):}{\textgreater}} luminositiesStellarAttributeMatch(<logical>[requireGettable] \to, <logical>[requireSettable] \to, <logical>[requireEvolvable] \to) Return a list of implementations that provide the given list off attributes for the luminositiesStellar property of the spheroid component.

tfamily \intzero luminositiesStellarCount() Compute the count of evolvable quantities in the luminositiesStellar property of the SpheroidStandard component.

tfamily \textcolor{red}{{\textless type(stellarLuminosities):}{\textgreater}} luminositiesStellarIsGettable() Get the luminositiesStellar property of the spheroid component.

tfamily \logicalzero luminositiesStellarIsSettable() Specify whether the luminositiesStellar property of the spheroid component is settable.

tfamily \void luminositiesStellarRate(<type(stellarLuminosities)> value) Cumulate to the rate of the luminositiesStellar property of the SpheroidStandard component.

tfamily \void luminositiesStellarScale(<type(stellarLuminosities)>value) Set the scale of the luminositiesStellar property of the SpheroidStandard component.

tfamily \void luminositiesStellarSet(<type(stellarLuminosities)> value) Set the luminositiesStellar property of the spheroid component.

tfamily \textcolor{red}{\textless double\textgreater} massGas() Get the massGas property of the spheroid component.

tfamily \textcolor{red}{{\textless type(varying\_string):}{\textgreater}} massGasAttributeMatch(<logical>[requireGettable] \to, <logical>[requireSettable] \to, <logical>[requireEvolvable] \to) Return a list of implementations that provide the given list off attributes for the massGas property of the spheroid component.

tfamily \intzero massGasCount() Compute the count of evolvable quantities in the massGas property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless double\textgreater} massGasIsGettable() Get the massGas property of the spheroid component.

tfamily \logicalzero massGasIsSettable() Specify whether the massGas property of the spheroid component is settable.

tfamily \void massGasRate(<double> value) Cumulate to the rate of the massGas property of the SpheroidStandard component.

tfamily \void massGasScale(<double> value) Set the scale of the massGas property of the SpheroidStandard component.

tfamily \void massGasSet(<double> value) Set the massGas property of the spheroid component.

tfamily \textcolor{red}{\textless type(varying\_string):}{\textgreater}} massGasSinkAttributeMatch(<logical>[requireGettable] \to, <logical>[requireSettable] \to, <logical>[requireEvolvable] \to) Return a list of implementations that provide the given list off attributes for the massGasSink property of the spheroid component.

tfamily \intzero massGasSinkCount() Compute the count of evolvable quantities in the massGasSink property of the SpheroidStandard component.
tfamily \textcolor{red}{\textless double\textgreater} massGasSinkIsGettable() Get the massGasSink property of the spheroid component.

tfamily \textcolor{red}{\textless double\textgreater} massGasSinkRate(<double> value) Cumulate to the rate of the massGasSink property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless double\textgreater} massStellar() Get the massStellar property of the spheroid component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(::)} massStellarAttributeMatch(<logical>[requireGettable] – > <logical>[requireSettable] – > <logical>[requireEvolvable] – >) Return a list of implementations that provide the given list off attributes for the massStellar property of the spheroid component.

tfamily \intzero massStellarCount() Compute the count of evolvable quantities in the massStellar property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless double\textgreater} massStellarIsGettable() Get the massStellar property of the spheroid component.

tfamily \logicalzero massStellarIsSettable() Specify whether the massStellar property of the spheroid component is settable.

tfamily \void massStellarRate(<double> value) Cumulate to the rate of the massStellar property of the SpheroidStandard component.

tfamily \void massStellarScale(<double> value) Set the scale of the massStellar property of the SpheroidStandard component.

tfamily \void massStellarSet(<double> value) Set the massStellar property of the spheroid component.

tfamily \void nameFromIndex(<integer> count – >, <varying_string>name – >) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the spheroid component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.


16.5. Objects

tfamily \doublezero potential{\textcolor{red}\textless double\textgreater\ radius\rightarrow, \textcolor{red}\textless componentType\textgreater\ [componentType]\rightarrow, \textcolor{red}\textless massType\textgreater\ [massType]\rightarrow, \textcolor{red}\textless logical\textgreater\ [haloLoaded]\rightarrow} Compute the gravitational potential.

tfamily \textcolor{red}\textless double\textgreater\ radius() Get the \textcolor{red}radius property of the \textcolor{red}spheroid component.

tfamily \textcolor{red}\textless type(varying\_\_string)(:\textgreater\ radiusAttributeMatch{\textcolor{red}\textless logical\textgreater\ [requireGettable]\rightarrow, \textcolor{red}\textless logical\textgreater\ [requireSettable]\rightarrow, \textcolor{red}\textless logical\textgreater\ [requireEvolvable]\rightarrow}) Return a list of implementations that provide the given list of attributes for the \textcolor{red}radius property of the \textcolor{red}spheroid component.

tfamily \textcolor{red}\textless double\textgreater\ radiusIsGettable() Get the \textcolor{red}radius property of the \textcolor{red}spheroid component.

\textcolor{red}\logicalzero radiusIsSettable() Specify whether the \textcolor{red}radius property of the \textcolor{red}spheroid component is settable.

tfamily \textcolor{red}\void\ radiusSet{\textcolor{red}\textless double\textgreater\ value} Set the \textcolor{red}radius property of the \textcolor{red}spheroid component.

tfamily \textcolor{red}\void\ readRaw{\textcolor{red}\textless integer\textgreater\ fileHandle\rightarrow} Read a binary dump of the \textcolor{red}nodeComponent from the given \textcolor{red}fileHandle.

tfamily \doublezero rotationCurve{\textcolor{red}\textless double\textgreater\ radius\rightarrow, \textcolor{red}\textless componentType\textgreater\ [componentType]\rightarrow, \textcolor{red}\textless massType\textgreater\ [massType]\rightarrow, \textcolor{red}\textless logical\textgreater\ [haloLoaded]\rightarrow} Compute the rotation curve.

tfamily \doublezero rotationCurveGradient{\textcolor{red}\textless double\textgreater\ radius\rightarrow, \textcolor{red}\textless componentType\textgreater\ [componentType]\rightarrow, \textcolor{red}\textless massType\textgreater\ [massType]\rightarrow, \textcolor{red}\textless logical\textgreater\ [haloLoaded]\rightarrow} Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \textcolor{red}\void\ serializeRates{\textcolor{red}\textless double(:)\textgreater\ array\rightarrow} Serialize the evolvable rates to an array.

tfamily \textcolor{red}\void\ serializeScales{\textcolor{red}\textless double(:)\textgreater\ array\rightarrow} Serialize the evolvable scales to an array.

tfamily \textcolor{red}\void\ serializeValues{\textcolor{red}\textless double(:)\textgreater\ array\rightarrow} Serialize the evolvable quantities to an array.

\textcolor{red}\logicalzero standardIsActive() Return whether the standard implementation of the \textcolor{red}spheroid component class is active.

\textcolor{red}\void\ starFormationHistoryRate{\textcolor{red}\textless type(history)\textgreater\ value} Cumulate to the rate of the \textcolor{red}starFormationHistory property of the \textcolor{red}SpheroidStandard component.

\textcolor{red}\intzero starFormationHistoryCount() Compute the count of evolvable quantities in the \textcolor{red}starFormationHistory property of the \textcolor{red}SpheroidStandard component.

\textcolor{red}\void\ starFormationHistoryIsGettable() Get the \textcolor{red}starFormationHistory property of the \textcolor{red}spheroid component.

\textcolor{red}\logicalzero starFormationHistoryIsSettable() Specify whether the \textcolor{red}starFormationHistory property of the \textcolor{red}spheroid component is settable.

\textcolor{red}\void\ starFormationHistoryRate{\textcolor{red}\textless type(history)\textgreater\ value} Cumulate to the rate of the \textcolor{red}starFormationHistory property of the \textcolor{red}SpheroidStandard component.
tfamily \ void \ starFormationHistoryScale(<type(history)> value) Set the scale of the \starFormationHistory property of the \SpheroidStandard component.

tfamily \ void \ starFormationHistorySet(<type(history)> value) Set the \starFormationHistory property of the \spheroid component.

tfamily \ textcolor{red}{\textless double\textgreater} starFormationRate() Get the \starFormationRate property of the \spheroid component.

tfamily \ textcolor{red}{\textless double\textgreater} starFormationRateAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the \starFormationRate property of the \spheroid component.

tfamily \ textcolor{red}{\textless double\textgreater} starFormationRateIsGettable() Get the \starFormationRate property of the \spheroid component.

tfamily \ logicalzero starFormationRateIsSettable() Specify whether the \starFormationRate property of the \spheroid component is settable.

tfamily \ textcolor{red}{\textless double\textgreater} stellarPropertiesHistory() Get the \stellarPropertiesHistory property of the \spheroid component.

tfamily \ textcolor{red}{\textless varying\_string\textgreater} stellarPropertiesHistoryAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the \stellarPropertiesHistory property of the \spheroid component.

tfamily \ intzero stellarPropertiesHistoryCount() Compute the count of evolvable quantities in the \stellarPropertiesHistory property of the \SpheroidStandard component.

tfamily \ textcolor{red}{\textless varying\_string\textgreater} type() Return the type of this object.

tfamily \ textcolor{red}{\textless double\textgreater} velocity() Get the \velocity property of the \spheroid component.
16.5. Objects

tfamily \textcolor{red}{\textless type(varying\_string)():\textgreater} velocityAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the velocity property of the spheroid component.

tfamily \textcolor{red}{\textlesstype(double):} velocityIsGettable() Get the velocity property of the spheroid component.

tfamily \logicalzero velocityIsSettable() Specify whether the velocity property of the spheroid component is settable.

tfamily \textcolor{red}{\textless type(double):} velocitySet(<double> value) Set the velocity property of the spheroid component.

tfamily nodeComponentSpheroidStandard

tfamily \textcolor{red}{\textless type(abundances):\textgreater} abundancesGas() Get the abundancesGas property of the spheroid component.

tfamily \textcolor{red}{\textlesstype(varying\_string):\textgreater} abundancesGasAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the abundancesGas property of the spheroid component.

tfamily \textcolor{red}{\textless type(abundances):\textgreater} abundancesGasIsGettable() Get the abundancesGas property of the spheroid component.

tfamily \logicalzero abundancesGasIsSettable() Specify whether the abundancesGas property of the spheroid component is settable.

tfamily \textcolor{red}{\textless type(abundances):\textgreater} abundancesGasRate(<type(abundances)> value) Cumulate to the rate of the abundancesGas property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless type(abundances):\textgreater} abundancesGasScale(<type(abundances)> value) Set the scale of the abundancesGas property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless type(abundances):\textgreater} abundancesGasSet(<type(abundances)> value) Set the abundancesGas property of the spheroid component.

tfamily \textcolor{red}{\textless type(abundances):\textgreater} abundancesStellar() Get the abundancesStellar property of the spheroid component.

tfamily \textcolor{red}{\textlesstype(varying\_string):\textgreater} abundancesStellarAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the abundancesStellar property of the spheroid component.

tfamily \textcolor{red}{\textless type(abundances):\textgreater} abundancesStellarCount() Compute the count of evolvable quantities in the abundancesStellar property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless type(abundances):\textgreater} abundancesStellarIsGettable() Get the abundancesStellar property of the spheroid component.

tfamily \logicalzero abundancesStellarIsSettable() Specify whether the abundancesStellar property of the spheroid component is settable.
tfamily \ void abundancesStellarRate(\textcolor{red}{\textless type(abundances)\textgreater} value) Cumulate to the rate of the abundancesStellar property of the SpheroidStandard component.

tfamily \ void abundancesStellarScale(\textcolor{red}{\textless type(abundances)\textgreater} value) Set the scale of the abundancesStellar property of the SpheroidStandard component.

tfamily \ void abundancesStellarSet(\textcolor{red}{\textless type(abundances)\textgreater} value) Set the abundancesStellar property of the spheroid component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentum() Get the angularMomentum property of the spheroid component.

tfamily \textcolor{red}{\textless type(varying\_string)(:\textgreater} angularMomentumAttributeMatch(\textcolor{red}{\textless logical\textgreater} [requireGettable] , \textcolor{red}{\textless logical\textgreater} [requireSettable] , \textcolor{red}{\textless logical\textgreater} [requireEvolvable] ) Return a list of implementations that provide the given list off attributes for the angularMomentum property of the spheroid component

tfamily \intzero angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless double\textgreater} angularMomentumIsGettable() Get the angularMomentum property of the spheroid component.

tfamily \logicalzero angularMomentumIsSettable() Specify whether the angularMomentum property of the spheroid component is settable.

tfamily \ void angularMomentumRate(\textcolor{red}{\textless double\textgreater} value) Cumulate to the rate of the angularMomentum property of the SpheroidStandard component.

tfamily \ void angularMomentumScale(\textcolor{red}{\textless double\textgreater} value) Set the scale of the angularMomentum property of the SpheroidStandard component.

tfamily \ void angularMomentumSet(\textcolor{red}{\textless double\textgreater} value) Set the angularMomentum property of the spheroid component.

tfamily \ void builder(\textcolor{red}{\textless type(node)\textgreater} componentDefinition\textcolor{red}{\rightarrow}) Build a nodeComponent from a supplied XML definition.

tfamily \ void createFunctionSet(\textcolor{red}{\textless function()\textgreater}) Set the function used to create SpheroidStandard components.

tfamily \ doublezero density(\textcolor{red}{\textless double(3)\textgreater} positionSpherical\textcolor{red}{\rightarrow}, \textcolor{red}{\textless componentType\textgreater} [componentType]\textcolor{red}{\rightarrow}, \textcolor{red}{\textless massType\textgreater} [massType]\textcolor{red}{\rightarrow}, \textcolor{red}{\textless weightBy\textgreater} [weightBy]\textcolor{red}{\rightarrow}, \textcolor{red}{\textless integer\textgreater} [weightIndex]\textcolor{red}{\rightarrow}, \textcolor{red}{\textless logical\textgreater} [haloLoaded]\textcolor{red}{\rightarrow}) Compute the density.

tfamily \ void deserializeRates(\textcolor{red}{\textless double(:)\textgreater} array\textcolor{red}{\leftarrow}) Deserialize the evolvable rates from an array.

tfamily \ void deserializeScales(\textcolor{red}{\textless double(:)\textgreater} array\textcolor{red}{\leftarrow}) Deserialize the evolvable scales from an array.

tfamily \ void deserializeValues(\textcolor{red}{\textless double(:)\textgreater} array\textcolor{red}{\leftarrow}) Deserialize the evolvable quantities from an array.

tfamily \ void destroy() Destroy the object.

tfamily \ void dump() Generate an ASCII dump of all properties.

tfamily \ void dumpRaw(\textcolor{red}{\textless integer\textgreater} fileHandle\textcolor{red}{\rightarrow}) Generate a binary dump of all properties.
tfamily \textcolor{red}{\textless \textbf{double}\textgreater} energyGasInput() Get the \textit{energyGasInput} property of the \textit{spheroid} component.

tfamily \textcolor{red}{\textless \textbf{string}\textgreater\textgreater} energyGasInputAttributeMatch() Return a list of implementations that provide the given list of attributes for the \textit{energyGasInput} property of the \textit{spheroid} component.

tfamily \textcolor{red}{\textless \textbf{double}\textgreater} energyGasInputIsGettable() Get the \textit{energyGasInput} property of the \textit{spheroid} component.

tfamily \textcolor{red}{\textbf{logical}} energyGasInputIsSettable() Specify whether the \textit{energyGasInput} property of the \textit{spheroid} component is settable.

tfamily \void energyGasInputRate(<\textbf{double}> value) Cumulate to the rate of the \textit{energyGasInput} property of the \textit{spheroid} component.

tfamily \void energyGasInputRateFunction(<\textbf{function}()> deferredFunction) Set the function to be used for the \textit{rate} method of the \textit{energyGasInput} property of the \textit{SpheroidStandard} component.

tfamily \textcolor{red}{\textbf{logical}} energyGasInputRateIsAttached() Return whether the rate method of the \textit{energyGasInput} property of the \textit{SpheroidStandard} component has been attached to a function.

tfamily \textcolor{red}{\textless \textbf{double}\textgreater} halfMassRadius() Get the \textit{halfMassRadius} property of the \textit{spheroid} component.

tfamily \textcolor{red}{\textless \textbf{string}\textgreater\textgreater} halfMassRadiusAttributeMatch() Return a list of implementations that provide the given list of attributes for the \textit{halfMassRadius} property of the \textit{spheroid} component.

tfamily \textcolor{red}{\textless \textbf{double}\textgreater} halfMassRadiusIsGettable() Get the \textit{halfMassRadius} property of the \textit{spheroid} component.

tfamily \textcolor{red}{\textbf{logical}} halfMassRadiusIsSettable() Specify whether the \textit{halfMassRadius} property of the \textit{spheroid} component is settable.

tfamily \textcolor{red}{\textbf{*type(treeNode)}} host() Return a pointer to the host \textit{treeNode} object.

tfamily \void hotHaloCoolingAbundancesRate(<\textbf{type(abundances)}> value) Cumulate to the rate of the \textit{hotHaloCoolingAbundances} property of the \textit{hotHalo} component.

tfamily \void hotHaloCoolingAngularMomentumRate(<\textbf{double}> value) Cumulate to the rate of the \textit{hotHaloCoolingAngularMomentum} property of the \textit{hotHalo} component.

tfamily \void hotHaloCoolingMassRate(<\textbf{double}> value) Cumulate to the rate of the \textit{hotHaloCoolingMass} property of the \textit{hotHalo} component.
tfamily \void initialize() Initialize the object.

tfamily \textcolor{red}{\textless logical\textgreater} isInitialized() Get the isInitialized property of the spheroid component.

tfamily \textcolor{red}{\textless logical\textgreater} isInitializedAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the isInitialized property of the spheroid component

tfamily \textcolor{red}{\textless logical\textgreater} isInitializedIsGettable() Get the isInitialized property of the spheroid component.

tfamily \logicalzero isInitializedIsSettable() Specify whether the isInitialized property of the spheroid component is settable.

tfamily \void isInitializedSet(<logical> value) Set the isInitialized property of the spheroid component.

tfamily \textcolor{red}{\textless type(stellarLuminosities)\textgreater} luminositiesStellar() Get the luminositiesStellar property of the spheroid component.

tfamily \textcolor{red}{\textless type(varying\_string):(\textgreater)} luminositiesStellarAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the luminositiesStellar property of the spheroid component

tfamily \intzero luminositiesStellarCount() Compute the count of evolvable quantities in the luminositiesStellar property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless double\textgreater} luminositiesStellarIsGettable() Get the luminositiesStellar property of the spheroid component.

tfamily \logicalzero luminositiesStellarIsSettable() Specify whether the luminositiesStellar property of the spheroid component is settable.

tfamily \void luminositiesStellarRate(<type(stellarLuminosities)> value) Cumulate to the rate of the luminositiesStellar property of the SpheroidStandard component.

tfamily \void luminositiesStellarScale(<type(stellarLuminosities)>value) Set the scale of the luminositiesStellar property of the SpheroidStandard component.

tfamily \void luminositiesStellarSet(<type(stellarLuminosities)> value) Set the luminositiesStellar property of the spheroid component.

tfamily \textcolor{red}{\textless double\textgreater} massGas() Get the massGas property of the spheroid component.

tfamily \textcolor{red}{\textless type(varying\_string):(\textgreater)} massGasAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the massGas property of the spheroid component

tfamily \intzero massGasCount() Compute the count of evolvable quantities in the massGas property of the SpheroidStandard component.

tfamily \textcolor{red}{\textless double\textgreater} massGasIsGettable() Get the massGas property of the spheroid component.
Objects

- **Objects:**
  - `tfamily logicalzero massGasIsSettable()` Specify whether the `massGas` property of the `spheroid` component is settable.
  - `tfamily void massGasRate(<double> value)` Cumulate to the rate of the `massGas` property of the `SpheroidStandard` component.
  - `tfamily void massGasScale(<double> value)` Set the scale of the `massGas` property of the `SpheroidStandard` component.
  - `tfamily void massGasSet(<double> value)` Set the `massGas` property of the `spheroid` component.
  - `tfamily textcolor{red}{<double> massGasSink() Get the `massGasSink` property of the `spheroid` component.}
  - `tfamily textcolor{red}{textless double{textgreater} massGasSinkAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the `massGasSink` property of the `spheroid` component.}
  - `tfamily intzero massGasSinkCount() Compute the count of evolvable quantities in the `massGasSink` property of the `SpheroidStandard` component.
  - `tfamily textcolor{red}{textless double{textgreater} massGasSinkIsGettable() Get the `massGasSink` property of the `spheroid` component.}
  - `tfamily logicalzero massGasSinkIsSettable()` Specify whether the `massGasSink` property of the `spheroid` component is settable.
  - `tfamily void massGasSinkRate(<double> value)` Cumulate to the rate of the `massGasSink` property of the `SpheroidStandard` component.
  - `tfamily void massGasSinkRateFunction(<function()> deferredFunction) Set the function to be used for the `rate` method of the `massGasSink` property of the `SpheroidStandard` component.
  - `tfamily logicalzero massGasSinkRateIsAttached() Return whether the rate method of the `massGasSink` property of the `SpheroidStandard` component has been attached to a function.
  - `tfamily textcolor{red}{<double> massStellar() Get the `massStellar` property of the `spheroid` component.}
  - `tfamily textcolor{red}{textless type(varying\_string)(::)textgreater} massStellarAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the `massStellar` property of the `spheroid` component.
  - `tfamily intzero massStellarCount() Compute the count of evolvable quantities in the `massStellar` property of the `SpheroidStandard` component.
  - `tfamily textcolor{red}{textless double{textgreater} massStellarIsGettable() Get the `massStellar` property of the `spheroid` component.}
  - `tfamily logicalzero massStellarIsSettable()` Specify whether the `massStellar` property of the `spheroid` component is settable.
  - `tfamily void massStellarRate(<double> value)` Cumulate to the rate of the `massStellar` property of the `SpheroidStandard` component.
16. Coding GALACTICUS

tfamily \void massStellarScale(<double> value) Set the scale of the massStellar property of the SpheroidStandard component.

tfamily \void massStellarSet(<double> value) Set the massStellar property of the spheroid component.

tfamily \void nameFromIndex(<integer> count →, <varying_string> name ←) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the spheroid component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

tfamily \void outputCount(<integer>integerPropertyCount ↔, <integer>doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

tfamily \void outputNames(<integer>integerProperty ↔, <char[*,:]>integerPropertyNames ↔, <char[*,:]>integerPropertyComments ↔, <double(:,:)>integerPropertyUnitsSI ↔, <integer>doubleProperty ↔, <char[*,:]>doublePropertyNames ↔, <char[*,:]>doublePropertyComments ↔, <double(:,:)>doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

tfamily \doublezeropotential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

tfamily \textcolor{red}{\textless double\textgreater} radius() Get the radius property of the spheroid component.

tfamily \textcolor{red}{\textless double\textgreater} radiusAttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the radius property of the spheroid component.

tfamily \textcolor{red}{\textless double\textgreater} radiusIsGettable() Get the radius property of the spheroid component.

tfamily \logicalzero radiusIsSettable() Specify whether the radius property of the spheroid component is settable.

tfamily \void radiusSet(<double> value) Set the radius property of the spheroid component.

tfamily \void readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezerorotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

tfamily \doublezerorotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.
16.5. Objects

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array) Serialize the evolvable quantities to an array.

tfamily \logicalzero standardIsActive() Return whether the standard implementation of the spheroid component class is active.

tfamily \textcolor{red}{\textless{} type(history)\textgreater{}} starFormationHistory() Get the \texttt{starFormationHistory} property of the \texttt{spheroid} component.

tfamily \textcolor{red}{\textless{} type(varying\_string)\textgreater{}} starFormationHistoryAttributeMatch(<logical>[requireGettable], <logical>[requireSettable], <logical>[requireEvolvable]) Return a list of implementations that provide the given list of attributes for the \texttt{starFormationHistory} property of the \texttt{spheroid} component.

tfamily \intzero starFormationHistoryCount() Compute the count of evolvable quantities in the \texttt{starFormationHistory} property of the \texttt{SpheroidStandard} component.

tfamily \textcolor{red}{\textless{} type(history)\textgreater{}} starFormationHistoryIsGettable() Get the \texttt{starFormationHistory} property of the \texttt{spheroid} component.

tfamily \logicalzero starFormationHistoryIsSettable() Specify whether the \texttt{starFormationHistory} property of the \texttt{spheroid} component is settable.

tfamily \void starFormationHistoryRate(<type(history)> value) Cumulate to the rate of the \texttt{starFormationHistory} property of the \texttt{SpheroidStandard} component.

tfamily \void starFormationHistoryScale(<type(history)> value) Set the scale of the \texttt{starFormationHistory} property of the \texttt{SpheroidStandard} component.

tfamily \void starFormationHistorySet(<type(history)> value) Set the \texttt{starFormationHistory} property of the \texttt{spheroid} component.

tfamily \textcolor{red}{\textless{} double\textgreater{}} starFormationRate() Get the \texttt{starFormationRate} property of the \texttt{spheroid} component.

tfamily \textcolor{red}{\textless{} type(varying\_string)\textgreater{}} starFormationRateAttributeMatch(<logical>[requireGettable], <logical>[requireSettable], <logical>[requireEvolvable]) Return a list of implementations that provide the given list of attributes for the \texttt{starFormationRate} property of the \texttt{spheroid} component.

tfamily \void starFormationRateFunction(<function()> deferredFunction) Set the function to be used for the \texttt{get} method of the \texttt{starFormationRate} property of the \texttt{SpheroidStandard} component.

tfamily \logicalzero starFormationRateIsAttached() Return whether the \texttt{get} method of the \texttt{starFormationRate} property of the \texttt{SpheroidStandard} component has been attached to a function.

tfamily \textcolor{red}{\textless{} double\textgreater{}} starFormationRateIsGettable() Get the \texttt{starFormationRate} property of the \texttt{spheroid} component.

tfamily \logicalzero starFormationRateIsSettable() Specify whether the \texttt{starFormationRate} property of the \texttt{spheroid} component is settable.
tfamily\textcolor{red}{\textless type(history)\textgreater} stellarPropertiesHistory() Get the stellarPropertiesHistory property of the \texttt{spheroid} component.

tfamily\textcolor{red}{\textless type(varying\_string)\textgreater} stellarPropertiesHistoryAttributeMatch(<logical>[requireGettable] \rightarrow, <logical>[requireSettable] \rightarrow, <logical>[requireEvolvable] \rightarrow) Return a list of implementations that provide the given list of attributes for the stellarPropertiesHistory property of the \texttt{spheroid} component.

tfamily\intzero stellarPropertiesHistoryCount() Compute the count of evolvable quantities in the stellarPropertiesHistory property of the \texttt{SpheroidStandard} component.

tfamily\textcolor{red}{\textless type(history)\textgreater} stellarPropertiesHistoryIsGettable() Get the stellarPropertiesHistory property of the \texttt{spheroid} component.

tfamily\textcolor{red}{\textless double\textgreater} velocity() Get the velocity property of the \texttt{spheroid} component.

672
tfamily\ void deserializeRates(<double(:)> array ←) Deserialize the evolvable rates from an array.

**Example:**
```cpp
// Deserialize evolvable rates
void deserializeRates(double[] rates);
```

**Implementation:**
```cpp
// C++ implementation
void deserializeRates(double[] rates) {
    // Deserialize rates from an array
}
```

**Description:**
- The `deserializeRates` method takes an array of doubles as input, which represents the evolvable rates of an object.
- It deserializes the rates from the given array.

### Other Deserialization Methods
- `deserializeScales(<double(:)> array ←)`
- `deserializeValues(<double(:)> array ←)`

**Destroy Method:**
- `void destroy()`

**Dump Methods:**
- `void dump()` Generate an ASCII dump of all properties.
- `void dumpRaw(<integer> fileHandle ←)` Generate a binary dump of all properties.
- `void dumpXML()` Generate an XML dump of all properties.

**Null Is Active Method:**
- `logical zero nullIsActive()`

**Initialize Method:**
- `void initialize()`

**Name From Index Method:**
- `void nameFromIndex(<integer> count →, <varying_string> name ←)`

**Ode Step Rates Initialize Method:**
- `void odeStepRatesInitialize()` Initialize rates for evolvable properties.

**Ode Step Scales Initialize Method:**
- `void odeStepScalesInitialize()` Initialize scales for evolvable properties.

**Output Method:**
- `void output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer> doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time →, <integer> instance →)` Generate values of outputtable properties.

**Output Count Method:**
- `void outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →)` Compute a count of outputtable properties.

**Output Names Method:**
- `void outputNames(<integer> integerProperty ↔, <char[*](:)> integerPropertyNames ↔, <char[*](::)> integerPropertyComments ↔, <double(:)> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char[*](::)> doublePropertyNames ↔, <char[*](::)> doublePropertyComments ↔, <double(:)> doublePropertyUnitsSI ↔, <double> time →, <integer> instance →)` Generate names of outputtable properties.

**Potential Method:**
- `double zero potential(<double> radius →, <componentType > [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →)`

**Example:**
```cpp
// Compute gravitational potential
double potential(double radius, componentType compType, massType massType, logical haloLoaded);
```
16. Coding **GALACTICUS**

tfamily \logicalzero preset3DIsActive() Return whether the preset3D implementation of the spin component class is active.

tfamily \logicalzero presetIsActive() Return whether the preset implementation of the spin component class is active.

tfamily \logicalzero randomIsActive() Return whether the random implementation of the spin component class is active.

tfamily \doublezero rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

tfamily \textcolor{red}{\textless double\textgreater} spin() Get the spin property of the spin component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)} spinAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the spin property of the spin component.

tfamily \intzero spinCount() Compute the count of evolvable quantities in the spin property of the SpinPreset component.

tfamily \textcolor{red}{\textless double\textgreater} spinGrowthRate() Get the spinGrowthRate property of the spin component.

tfamily \textcolor{red}{\textlesstype(varying\_string)(:)} spinGrowthRateAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the spinGrowthRate property of the spin component.

tfamily \textcolor{red}{\textless double\textgreater} spinGrowthRateIsGettable() Get the spinGrowthRate property of the spin component.

tfamily \logicalzero spinGrowthRateIsSettable() Specify whether the spinGrowthRate property of the spin component is settable.

tfamily \void spinGrowthRateSet(<double> value) Set the spinGrowthRate property of the spin component.

tfamily \textcolor{red}{\textless double\textgreater} spinIsGettable() Get the spin property of the spin component.

tfamily \logicalzero spinIsSettable() Specify whether the spin property of the spin component is settable.

tfamily \void spinRate(<double> value) Cumulate to the rate of the spin property of the SpinPreset component.
16.5. Objects

tfamily \void spinScale(<double> value) Set the scale of the \texttt{spin} property of the \texttt{SpinPreset} component.

tfamily \void spinSet(<double> value) Set the \texttt{spin} property of the \texttt{spin} component.

tfamily \texttt{spinVector()} Get the \texttt{spinVector} property of the \texttt{spin} component.

tfamily \texttt{spinVectorAttributeMatch(<logical>[requireGettable\rightarrow, <logical>[requireSettable\rightarrow, <logical>[requireEvolvable\rightarrow]) Return a list of implementations that provide the given list of attributes for the \texttt{spinVector} property of the \texttt{spin} component.

tfamily \intzero spinVectorCount() Compute the count of evolvable quantities in the \texttt{spinVector} property of the \texttt{SpinPreset3D} component.

tfamily \texttt{spinVectorGrowthRate()} Get the \texttt{spinVectorGrowthRate} property of the \texttt{spin} component.

tfamily \texttt{spinVectorGrowthRateAttributeMatch(<logical>[requireGettable\rightarrow, <logical>[requireSettable\rightarrow, <logical>[requireEvolvable\rightarrow]) Return a list of implementations that provide the given list of attributes for the \texttt{spinVectorGrowthRate} property of the \texttt{spin} component.

tfamily \texttt{spinVectorGrowthRateIsGettable()} Get the \texttt{spinVectorGrowthRate} property of the \texttt{spin} component.

tfamily \logicalzero spinVectorGrowthRateIsSettable() Specify whether the \texttt{spinVectorGrowthRate} property of the \texttt{spin} component is settable.

tfamily \void spinVectorGrowthRateSet(<double(:)> value) Set the \texttt{spinVectorGrowthRate} property of the \texttt{spin} component.

tfamily \texttt{spinVectorIsGettable()} Get the \texttt{spinVector} property of the \texttt{spin} component.

tfamily \logicalzero spinVectorIsSettable() Specify whether the \texttt{spinVector} property of the \texttt{spin} component is settable.

tfamily \void spinVectorRate(<double(:)> value) Cumulate to the rate of the \texttt{spinVector} property of the \texttt{SpinPreset3D} component.

tfamily \void spinVectorScale(<double(:)> value) Set the scale of the \texttt{spinVector} property of the \texttt{SpinPreset3D} component.

tfamily \void spinVectorSet(<double(:)> value) Set the \texttt{spinVector} property of the \texttt{spin} component.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical\rightarrow, <componentType>[componentType\rightarrow, <massType>[massType\rightarrow, <weightBy>[weightBy\rightarrow, <integer>[weightIndex\rightarrow, <logical>[haloLoaded\rightarrow]) Compute the surface density.

tfamily \texttt{type()} Return the type of this object.
tfamily nodeComponentSpinNull

tfamily \ void builder(\*type(node)>componentDefinition\rightarrow) Build a nodeComponent from a supplied XML definition.

tfamily \ double zero\ density(\*double(3)> positionSpherical\rightarrow, \*componentType \ [\text{componentType}\rightarrow, \*massType \ [\text{massType}\rightarrow, \*weightBy \ [\text{weightBy}\rightarrow, \*integer \ [\text{weightIndex}\rightarrow, \*logical > \text{haloLoaded}]\rightarrow) Compute the density.

tfamily \ void\ deserializeRates(\*double(:)> array\leftarrow) Deserialize the evolvable rates from an array.

tfamily \ void\ deserializeScales(\*double(:)> array\leftarrow) Deserialize the evolvable scales from an array.

tfamily \ void\ deserializeValues(\*double(:)> array\leftarrow) Deserialize the evolvable quantities from an array.

tfamily \ void destroy() Destroy the object.

tfamily \ void dump() Generate an ASCII dump of all properties.

tfamily \ void\ dumpRaw(\*integer> fileHandle\rightarrow) Generate a binary dump of all properties.

tfamily \ void\ dumpXML() Generate an XML dump of all properties.

tfamily \ double zero\ enclosedMass(\*double> radius\rightarrow, \*componentType \ [\text{componentType}\rightarrow, \*massType \ [\text{massType}\rightarrow, \*weightBy \ [\text{weightBy}\rightarrow, \*integer \ [\text{weightIndex}\rightarrow, \*logical > \text{haloLoaded}]\rightarrow) Compute the mass enclosed within a radius.

tfamily \ textcolor{red}{\textless type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \ void\ hotHaloCoolingAbundancesRate(\*type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \ void\ hotHaloCoolingAngularMomentumRate(\*double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \ void\ hotHaloCoolingMassRate(\*double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \ void initialize() Initialize the object.

tfamily \ void\ nameFromIndex(\*integer> count\rightarrow, \*varying_string> name\leftarrow) Return the name of a property given is index.

tfamily \ logical zero nullIsActive() Return whether the null implementation of the spin component class is active.

tfamily \ void\ odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \ void\ odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \ void\ output(\*integer> integerProperty\leftarrow, \*integer> integerBufferCount\leftarrow, \*integer(\*):> integerBuffer\leftarrow, \*integer> doubleProperty\leftarrow, \*integer> doubleBufferCount\leftarrow, \*double(\*):> doubleBuffer\leftarrow, \*double> time\rightarrow, \*integer> instance\rightarrow) Generate values of outputtable properties.

tfamily \ void\ outputCount(\*integer> integerPropertyCount\leftarrow, \*integer> doublePropertyCount\leftarrow, \*double> time\rightarrow, \*integer> instance\rightarrow) Compute a count of outputtable properties.
16.5. Objects

tfamily void outputNames(<integer> integerProperty, <char[*]) integerPropertyNames, <char[*]) integerPropertyComments, <double()> integerPropertyUnitsSI, <integer> doubleProperty, <char[*]) doublePropertyNames, <char[*]) doublePropertyComments, <double()> doublePropertyUnitsSI, <double>() time, <integer>() instance) Generate names of outputtable properties.

tfamily doublezero potential(<double>() radius, <componentType>() componentType, <massType>() massType, <logical> haloLoaded) Compute the gravitational potential.

tfamily logicalzero preset3DIsActive() Return whether the preset3D implementation of the spin component class is active.

tfamily logicalzero presetIsActive() Return whether the preset implementation of the spin component class is active.

tfamily logicalzero randomIsActive() Return whether the random implementation of the spin component class is active.

tfamily void readRaw(<integer>() fileHandle) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily doublezero rotationCurve(<double>() radius, <componentType>() componentType, <massType>() massType, <logical> haloLoaded) Compute the rotation curve.

tfamily doublezero rotationCurveGradient(<double>() radius, <componentType>() componentType, <massType>() massType, <logical> haloLoaded) Compute the rotation curve gradient.

tfamily intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily void serializeRates(<double>() array) Serialize the evolvable rates to an array.

tfamily void serializeScales(<double>() array) Serialize the evolvable scales to an array.

tfamily void serializeValues(<double>() array) Serialize the evolvable quantities to an array.

tfamily textcolor{red}{} textless double{} textgreater{} spin() Get the spin property of the spin component.

tfamily textcolor{red}{} textless type(varying\_string()){} textgreater{} spinAttributeMatch(<logical>() requireGettable, <logical>() requireSettable, <logical>() requireEvolvable) Return a list of implementations that provide the given list of attributes for the spin property of the spin component.

tfamily intzero spinCount() Compute the count of evolvable quantities in the spin property of the SpinPreset component.

tfamily textcolor{red}{} textless double{} textgreater{} spinGrowthRate() Get the spinGrowthRate property of the spin component.

tfamily textcolor{red}{} textless type(varying\_string()){} textgreater{} spinGrowthRateAttributeMatch(<logical>() requireGettable, <logical>() requireSettable, <logical>() requireEvolvable) Return a list of implementations that provide the given list of attributes for the spinGrowthRate property of the spin component.

tfamily textcolor{red}{} textless double{} textgreater{} spinGrowthRateIsGettable() Get the spinGrowthRate property of the spin component.
tfamily \logicalzero spinGrowthRateIsSettable() Specify whether the spinGrowthRate property of the spin component is settable.

tfamily \void spinGrowthRateSet(<double> value) Set the spinGrowthRate property of the spin component.

tfamily \textcolor{red}\textless double\textgreater \isGettable() Get the spin property of the spin component.

tfamily \logicalzero spinIsSettable() Specify whether the spin property of the spin component is settable.

tfamily \void spinRate(<double> value) Cumulate to the rate of the spin property of the SpinPreset component.

tfamily \void spinScale(<double> value) Set the scale of the spin property of the SpinPreset component.

tfamily \void spinSet(<double> value) Set the spin property of the spin component.

\textcolor{red}\textless double\textgreater \textgreater \isGettable() Get the spinVector property of the spin component.

tfamily \textcolor{red}\textless type(varying\_string)(\_):\textgreater \AttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the spinVector property of the spin component.

tfamily \intzero spinVectorCount() Compute the count of evolvable quantities in the spinVector property of the SpinPreset3D component.

\textcolor{red}\textless double\textgreater \isGettable() Get the spinVector property of the spin component.

\textcolor{red}\textless type(varying\_string)(\_):\textgreater \AttributeMatch(<logical>[requireGettable] →, <logical>[requireSettable] →, <logical>[requireEvolvable] →) Return a list of implementations that provide the given list of attributes for the spinVectorGrowthRate property of the spin component.

\textcolor{red}\textless double\textgreater \isGettable() Get the spinVector property of the spin component.

\logicalzero spinVectorGrowthRateIsSettable() Specify whether the spinVectorGrowthRate property of the spin component is settable.

\void spinVectorGrowthRateSet(<double>:value) Set the spinVectorGrowthRate property of the spin component.

\textcolor{red}\textless double\textgreater \isGettable() Get the spinVector property of the spin component.

\logicalzero spinVectorIsSettable() Specify whether the spinVector property of the spin component is settable.

\void spinVectorRate(<double>:value) Cumulate to the rate of the spinVector property of the SpinPreset3D component.
16.5. Objects

tfamily \void spinVectorScale(<double>(); value) Set the scale of the spinVector property of the SpinPreset3D component.

tfamily \void spinVectorSet(<double>(); value) Set the spinVector property of the spin component.

tfamily \doublezero surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.

tfamily \textcolor{red}{\textless type(varying\_string)\textgreater} type() Return the type of this object.

tfamily nodeComponentSpinPreset

tfamily \void builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily \doublezero density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double>(); array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double>(); array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double>(); array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily \void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count→, <varying_string> name←) Return the name of a property given is index.
tfamily \logicalzero nullIsActive() Return whether the null implementation of the spin component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty\leftrightarrow, <integer> integerBufferCount\leftrightarrow, <integer(:,:)> integerBuffer\leftrightarrow, <integer>doubleProperty\leftrightarrow, <integer> doubleBufferCount\leftrightarrow, <double(:,:)> doubleBuffer\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow) Generate values of outputtable properties.

tfamily \void outputCount(<integer>integerPropertyCount\leftrightarrow, <integer> doublePropertyCount\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty\leftrightarrow, <char\[*](:)> integerPropertyNames\leftrightarrow, <char\[*](:)> integerPropertyComments\leftrightarrow, <double(:)> integerPropertyUnitsSI\leftrightarrow, <integer> doubleProperty\leftrightarrow, <char\[*](:)> doublePropertyNames\leftrightarrow, <char\[*](:)> doublePropertyComments\leftrightarrow, <double(:)> doublePropertyUnitsSI\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow) Generate names of outputtable properties.

tfamily \doublezero potential(<double> radius\rightarrow, <componentType> [componentType]\rightarrow, <massType> [massType]\rightarrow, <logical> [haloLoaded]\rightarrow) Compute the gravitational potential.

tfamily \logicalzero preset3DIsActive() Return whether the preset3D implementation of the spin component class is active.

tfamily \logicalzero presetIsActive() Return whether the preset implementation of the spin component class is active.

tfamily \logicalzero randomIsActive() Return whether the random implementation of the spin component class is active.

tfamily \void readRaw(<integer> fileHandle\rightarrow) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \doublezero rotationCurve(<double> radius\rightarrow, <componentType> [componentType]\rightarrow, <massType> [massType]\rightarrow, <logical> [haloLoaded]\rightarrow) Compute the rotation curve.

tfamily \doublezero rotationCurveGradient(<double> radius\rightarrow, <componentType> [componentType]\rightarrow, <massType> [massType]\rightarrow, <logical> [haloLoaded]\rightarrow) Compute the rotation curve gradient.

tfamily \intzero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double(:)> array\rightarrow) Serialize the evolvable rates to an array.

tfamily \void serializeScales(<double(:)> array\rightarrow) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array\rightarrow) Serialize the evolvable quantities to an array.

tfamily \textcolor{red}{\textless double\textgreater} spin() Get the spin property of the spin component.

tfamily \textcolor{red}{\textless double\textgreater} spinAttributeMatch(<logical>[requireGettable]\rightarrow, <logical>[requireSettable]\rightarrow, <logical>[requireEvolvable]\rightarrow) Return a list of implementations that provide the given list off attributes for the spin property of the spin component.
16.5. Objects

fam styl \textcolor{red}{\textless double\textgreater} spinCount() Compute the count of evolvable quantities in the spin property of the SpinPreset component.

fam styl \textcolor{red}{\textless double\textgreater} spinGrowthRate() Get the spinGrowthRate property of the spin component.

fam styl \textcolor{red}{\textless double\textgreater} spinGrowthRateIsGettable() Get the spinGrowthRate property of the spin component.

fam styl \textcolor{red}{\textless double\textgreater} spinGrowthRateIsSettable() Specify whether the spinGrowthRate property of the spin component is settable.

fam styl \int zero spinIsGettable() Get the spin property of the spin component.

fam styl \int zero spinIsSettable() Specify whether the spin property of the spin component is settable.

fam styl \void spinRate(<double> value) Cumulate to the rate of the spin property of the SpinPreset component.

fam styl \void spinScale(<double> value) Set the scale of the spin property of the SpinPreset component.

fam styl \int zero spinSet(<double> value) Set the spin property of the spin component.

fam styl \textcolor{red}{\textless double\textgreater} spinVector() Get the spinVector property of the spin component.

fam styl \textcolor{red}{\textless double\textgreater} spinVectorGrowthRate() Get the spinVectorGrowthRate property of the spin component.

fam styl \textcolor{red}{\textless double\textgreater} spinVectorGrowthRateIsGettable() Get the spinVectorGrowthRate property of the spin component.

fam styl \textcolor{red}{\textless double\textgreater} spinVectorGrowthRateIsSettable() Specify whether the spinVectorGrowthRate property of the spin component is settable.

fam styl \int zero spinVectorCount() Compute the count of evolvable quantities in the spinVector property of the SpinPreset3D component.

fam styl \textcolor{red}{\textless double\textgreater} spinVectorGrowthRate() Get the spinVectorGrowthRate property of the spin component.

fam styl \textcolor{red}{\textless double\textgreater} spinVectorGrowthRateIsGettable() Get the spinVectorGrowthRate property of the spin component.

fam styl \textcolor{red}{\textless double\textgreater} spinVectorGrowthRateIsSettable() Specify whether the spinVectorGrowthRate property of the spin component is settable.
16. Coding Galacticus

tfamily \textcolor{red}{\textless type(treeNode)\textgreater} host() Return a pointer to the host treeNode object.

tfamily void hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.
tfamily \void hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(<integer> count \rightarrow, <varying_string> name \leftarrow) Return the name of a property given is index.

tfamily \logical zero nullIsActive() Return whether the null implementation of the spin component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(<integer> integerProperty \leftrightarrow, <integer> integerBufferCount \leftrightarrow, <integer[;,:]> integerBuffer \leftrightarrow, <integer> doubleProperty \leftrightarrow, <integer> doubleBufferCount \leftrightarrow, <double[;,:]> doubleBuffer \leftrightarrow, <double> time \rightarrow, <integer> instance \rightarrow) Generate values of outputtable properties.

tfamily \void outputCount(<integer> integerPropertyCount \leftrightarrow, <integer> doublePropertyCount \leftrightarrow, <double> time \rightarrow, <integer> instance \rightarrow) Compute a count of outputtable properties.

tfamily \void outputNames(<integer> integerProperty \leftrightarrow, <char[*]> integerPropertyNames \leftrightarrow, <char[*]> integerPropertyComments \leftrightarrow, <double[*]> integerPropertyUnitsSI \leftrightarrow, <integer> doubleProperty \leftrightarrow, <char[*]> doublePropertyNames \leftrightarrow, <char[*]> doublePropertyComments \leftrightarrow, <double[*]> doublePropertyUnitsSI \leftrightarrow, <double> time \rightarrow, <integer> instance \rightarrow) Generate names of outputtable properties.

tfamily \double zero potential(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the gravitational potential.

tfamily \logical zero preset3DIsActive() Return whether the preset3D implementation of the spin component class is active.

tfamily \logical zero presetIsActive() Return whether the preset implementation of the spin component class is active.

tfamily \logical zero randomIsActive() Return whether the random implementation of the spin component class is active.

tfamily \void readRaw(<integer> fileHandle \rightarrow) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \double zero rotationCurve(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the rotation curve.

tfamily \double zero rotationCurveGradient(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow) Compute the rotation curve gradient.

\int zero serializeCount() Return a count of the number of evolvable quantities to be evolved.

tfamily \void serializeRates(<double[;,:]> array \rightarrow) Serialize the evolvable rates to an array.
16. Coding GALACTICUS

tfamily \void serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

tfamily \void serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

tfamily \textcolor{red}{\textless double\textgreater} spin() Get the spin property of the spin component.

tfamily \textcolor{red}{\textlesstype(varying_string)(:)} spinAttributeMatch(<logical>[requireGettable]→,
<logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the spin property of the spin component.

tfamily \intzero spinCount() Compute the count of evolvable quantities in the spin property of the SpinPreset component.

tfamily \textcolor{red}{\textless double\textgreater} spinGrowthRate() Get the spinGrowthRate property of the spin component.

tfamily \textcolor{red}{\textlesstype(varying_string)(:)} spinGrowthRateAttributeMatch(<logical>[requireGettable]→,
<logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the spinGrowthRate property of the spin component.

tfamily \textcolor{red}{\textless double\textgreater} spinGrowthRateIsGettable() Get the spinGrowthRate property of the spin component.

tfamily \logicalzero spinGrowthRateIsSettable() Specify whether the spinGrowthRate property of the spin component is settable.

tfamily \void spinGrowthRateSet(<double> value) Set the spinGrowthRate property of the spin component.

tfamily \textcolor{red}{\textless double\textgreater} spinIsGettable() Get the spin property of the spin component.

tfamily \logicalzero spinIsSettable() Specify whether the spin property of the spin component is settable.

tfamily \void spinRate(<double> value) Cumulate to the rate of the spin property of the SpinPreset component.

tfamily \void spinScale(<double> value) Set the scale of the spin property of the SpinPreset component.

tfamily \void spinSet(<double> value) Set the spin property of the spin component.

tfamily \textcolor{red}{\textless double(:)} spinVector() Get the spinVector property of the spin component.

tfamily \textcolor{red}{\textlesstype(varying_string)(:)} spinVectorAttributeMatch(<logical>[requireGettable]→,
<logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list off attributes for the spinVector property of the spin component.

tfamily \intzero spinVectorCount() Compute the count of evolvable quantities in the spinVector property of the SpinPreset3D component.

tfamily \textcolor{red}{\textless double(:)} spinVectorGrowthRate() Get the spinVectorGrowthRate property of the spin component.
16.5. Objects

tfamily \textcolor{red}{\textless double():\textgreater} spinVectorGrowthRateIsGettable() Get the spinVectorGrowthRate property of the spin component.

tfamily \textcolor{red}{\textless double():\textgreater} spinVectorGrowthRateIsSettable() Specify whether the spinVectorGrowthRate property of the spin component is settable.

tfamily \void spinVectorGrowthRateSet(<double(:)> value) Set the spinVectorGrowthRate property of the spin component.

tfamily \textcolor{red}{\textless double():\textgreater} spinVectorIsGettable() Get the spinVector property of the spin component.

tfamily \logiczero spinVectorIsSettable() Specify whether the spinVector property of the spin component is settable.

tfamily \void spinVectorRate(<double(:)> value) Cumulate to the rate of the spinVector property of the SpinPreset3D component.

tfamily \void spinVectorScale(<double(:)> value) Set the scale of the spinVector property of the SpinPreset3D component.

tfamily \void spinVectorSet(<double(:)> value) Set the spinVector property of the spin component.

tfamily \double zero surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the surface density.

tfamily \textcolor{red}{\textless type(varying\_string):\textgreater} type() Return the type of this object.

nodeComponentSpinRandom

tfamily \void builder(<\texttt{type(node)}>componentDefinition→) Build a nodeComponent from a supplied XML definition.

tfamily \double zero density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

tfamily \void deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.

tfamily \void deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.

tfamily \void deserializeValues(<double(:)> array←) Deserialize the evolvable quantities from an array.

tfamily \void destroy() Destroy the object.

tfamily \void dump() Generate an ASCII dump of all properties.

tfamily \void dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.
16. Coding GALACTICUS

tfamily \void dumpXML() Generate an XML dump of all properties.

tfamily \doublezero enclosedMass(\double radius\rightarrow, \componentType componentType\rightarrow, \massType massType\rightarrow, \weightBy weightBy\rightarrow, \integer weightIndex\rightarrow, \logical haloLoaded\rightarrow) Compute the mass enclosed within a radius.

tfamily \textcolor{red}{\textless *type(treeNode)\textgreater} host() Return a pointer to the host *treeNode* object.

tfamily \void hotHaloCoolingAbundancesRate(\type(abundances) value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

tfamily \void hotHaloCoolingAngularMomentumRate(\double value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

tfamily \void hotHaloCoolingMassRate(\double value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

tfamily \void initialize() Initialize the object.

tfamily \void nameFromIndex(\integer count\rightarrow, \varying_string name\leftarrow) Return the name of a property given is index.

tfamily \logicalzero nullIsActive() Return whether the null implementation of the spin component class is active.

tfamily \void odeStepRatesInitialize() Initialize rates for evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize scales for evolvable properties.

tfamily \void output(\integer integerProperty\leftrightarrow, \integer integerBufferCount\leftrightarrow, \integer(:;) integerBuffer\leftrightarrow, \integer doubleProperty\leftrightarrow, \integer doubleBufferCount\leftrightarrow, \double(:;) doubleBuffer\leftrightarrow, \double time\rightarrow, \integer instance\rightarrow) Generate values of outputtable properties.

tfamily \void outputCount(\integer integerPropertyCount\leftrightarrow, \integer doublePropertyCount\leftrightarrow, \double time\rightarrow, \integer instance\rightarrow) Compute a count of outputtable properties.

tfamily \void outputNames(\integer integerProperty\leftrightarrow, \char[*](:) integerPropertyNames\leftrightarrow, \char[*](:) integerPropertyComments\leftrightarrow, \double(:) integerPropertyUnitsSI\leftrightarrow, \integer doubleProperty\leftrightarrow, \char[*](:) doublePropertyNames\leftrightarrow, \char[*](:) doublePropertyComments\leftrightarrow, \double(:) doublePropertyUnitsSI\leftrightarrow, \double time\rightarrow, \integer instance\rightarrow) Generate names of outputtable properties.

tfamily \doublezero potential(\double radius\rightarrow, \componentType componentType\rightarrow, \massType massType\rightarrow, \logical haloLoaded\rightarrow) Compute the gravitational potential.

tfamily \logicalzero preset3DIsActive() Return whether the preset3D implementation of the spin component class is active.

tfamily \logicalzero presetIsActive() Return whether the preset implementation of the spin component class is active.

tfamily \logicalzero randomIsActive() Return whether the random implementation of the spin component class is active.
tfamily \**\textfamilydefault\void\**\txtfamilydefault{readRaw}(\<\text{integer}\> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

tfamily \**\textfamilydefault{doublezero}\**\txtfamilydefault{rotationCurve}(\<\text{double}\> radius →, \<\text{componentType}\> [componentType] →, \<\text{massType}\> [massType] →, \<\text{logical}\> [haloLoaded] →) Compute the rotation curve.

tfamily \**\textfamilydefault{doublezero}\**\txtfamilydefault{rotationCurveGradient}(\<\text{double}\> radius →, \<\text{componentType}\> [componentType] →, \<\text{massType}\> [massType] →, \<\text{logical}\> [haloLoaded] →) Compute the rotation curve gradient.

tfamily \**\textfamilydefault{intzero} serializeCount()\** Return a count of the number of evolvable quantities to be evolved.

tfamily \**\textfamilydefault{void} serializeRates(\<\text{double}(:)\> array →)\** Serialize the evolvable rates to an array.

tfamily \**\textfamilydefault{void} serializeScales(\<\text{double}(:)\> array →)\** Serialize the evolvable scales to an array.

tfamily \**\textfamilydefault{void} serializeValues(\<\text{double}(:)\> array →)\** Serialize the evolvable quantities to an array.

tfamily \**\textcolor{red}{{\text{double}\textgreater}}\**\txtfamilydefault{spin()}\** Get the spin property of the spin component.

tfamily \**\textcolor{red}{{\text{type(varying_string)(:)}\textgreater}}\**\txtfamilydefault{spinAttributeMatch}(\<\text{logical}\> [requireGettable] →, \<\text{logical}\> [requireSettable] →, \<\text{logical}\> [requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the spin property of the spin component

tfamily \**\textfamilydefault{intzero} spinCount()\** Compute the count of evolvable quantities in the spin property of the SpinPreset component.

tfamily \**\textcolor{red}{{\text{double}\textgreater}}\**\txtfamilydefault{spinGrowthRate()}\** Get the spinGrowthRate property of the spin component.

tfamily \**\textcolor{red}{{\text{type(varying_string)(:)}\textgreater}}\**\txtfamilydefault{spinGrowthRateAttributeMatch}(\<\text{logical}\> [requireGettable] →, \<\text{logical}\> [requireSettable] →, \<\text{logical}\> [requireEvolvable] →) Return a list of implementations that provide the given list off attributes for the spinGrowthRate property of the spin component

tfamily \**\textcolor{red}{{\text{double}\textgreater}}\**\txtfamilydefault{spinGrowthRateIsGettable()}\** Get the spinGrowthRate property of the spin component.

tfamily \**\textfamilydefault{logicalzero} spinGrowthRateIsSettable()\** Specify whether the spinGrowthRate property of the spin component is settable.

tfamily \**\textfamilydefault{void} spinGrowthRateSet(\<\text{double}\> value)\** Set the spinGrowthRate property of the spin component.

tfamily \**\textcolor{red}{{\text{double}\textgreater}}\**\txtfamilydefault{spinIsGettable()}\** Get the spin property of the spin component.

tfamily \**\textfamilydefault{logicalzero} spinIsSettable()\** Specify whether the spin property of the spin component is settable.

tfamily \**\textfamilydefault{void} spinRate(\<\text{double}\> value)\** Cumulate to the rate of the spin property of the SpinPreset component.

tfamily \**\textfamilydefault{void} spinScale(\<\text{double}\> value)\** Set the scale of the spin property of the SpinPreset component.

tfamily \**\textfamilydefault{void} spinSet(\<\text{double}\> value)\** Set the spin property of the spin component.
tfamily \textcolor{red}{\textless textless double(:)\textgreater} \spinVector() Get the \textbf{spinVector} property of the \textbf{spin} component.

tfamily \textcolor{red}{\textless textless type(varying\_string)(:)\textgreater} \spinVectorAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the \textbf{spinVector} property of the \textbf{spin} component.

tfamily \textcolor{red}{\textless double(:)\textgreater} \spinVectorCount() Compute the count of evolvable quantities in the \textbf{spinVector} property of the \textbf{SpinPreset3D} component.

tfamily \textcolor{red}{\textless double(:)\textgreater} \spinVectorGrowthRate() Get the \textbf{spinVectorGrowthRate} property of the \textbf{spin} component.

tfamily \textcolor{red}{\textless double(:)\textgreater} \spinVectorGrowthRateAttributeMatch(<logical>[requireGettable]→, <logical>[requireSettable]→, <logical>[requireEvolvable]→) Return a list of implementations that provide the given list of attributes for the \textbf{spinVectorGrowthRate} property of the \textbf{spin} component.

tfamily \textcolor{red}{\textless double(:)\textgreater} \spinVectorGrowthRateIsGettable() Get the \textbf{spinVectorGrowthRate} property of the \textbf{spin} component.

tfamily \textcolor{red}{\textless double(:)\textgreater} \spinVectorGrowthRateIsSettable() Specify whether the \textbf{spinVectorGrowthRate} property of the \textbf{spin} component is settable.

tfamily \textcolor{red}{\textless double(:)\textgreater} \spinVectorGrowthRateSet(<double(:)> value) Set the \textbf{spinVectorGrowthRate} property of the \textbf{spin} component.

tfamily \textcolor{red}{\textless textless double(:)\textgreater} \spinVectorIsGettable() Get the \textbf{spinVector} property of the \textbf{spin} component.

tfamily \textcolor{red}{\textless logical\textgreater} \spinVectorIsSettable() Specify whether the \textbf{spinVector} property of the \textbf{spin} component is settable.

tfamily \textcolor{red}{\textless double(:)\textgreater} \spinVectorRate(<double(:)> value) Cumulate to the rate of the \textbf{spinVector} property of the \textbf{SpinPreset3D} component.

tfamily \textcolor{red}{\textless double(:)\textgreater} \spinVectorScale(<double(:)> value) Set the scale of the \textbf{spinVector} property of the \textbf{SpinPreset3D} component.

tfamily \textcolor{red}{\textless double(:)\textgreater} \spinVectorSet(<double(:)> value) Set the \textbf{spinVector} property of the \textbf{spin} component.

tfamily \textcolor{red}{\textless doublearf\_3>(positionCylindrical→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the surface density.

tfamily \textcolor{red}{\textless textless type(varying\_string)\textgreater} \textbf{type}() Return the type of this object.

tfamily progenitorIterator

   tfamily \textcolor{red}{\textless textless *class(nodeData)\textgreater} \textbf{current}(<class(nodeData)(:)> nodes→) Return a pointer to the current progenitor.

tfamily \textcolor{red}{\textless class(nodeData)\textgreater} \textbf{descendentSet}(<class(nodeData)> node→, <class(nodeData)(:)> nodes→) Set the target descendent node and initialize the iterator.
16.5. Objects

tfamily \logicalzero exist() Return true if any progenitors exist, false otherwise.

tfamily \textcolor{red}{\textless integer(kind=kind\_int8)\textgreater} index(<class(nodeData)(:)> nodes→) Return the index of the current progenitor.

tfamily \logicalzero next(<class(nodeData)(:)> nodes→) Move to the next progenitor. Returns true if the next progenitor exists, false otherwise.

tfamily radiationStructure

tfamily \void define(<integer(:)> radiationTypes→) Define the radiation components active in a given radiation object.

tfamily \doublezero flux(<double> wavelength→, <integer(:)> [radiationType]→) Return the flux (in units of ergs cm\(^2\) s\(^{-1}\) Hz\(^{-1}\) ster\(^{-1}\)) of the given radiation structure.

tfamily \doublezero integrateOverCrossSection(<double> wavelength→, <integer(:)> [radiationType]→) Integrates the flux (in units of ergs cm\(^2\) s\(^{-1}\) Hz\(^{-1}\) ster\(^{-1}\)) of the given radiation structure between the wavelengths given in wavelengthRange over a cross section specified by the function crossSectionFunction.

tfamily \logicalzero isDefined() Return true if the radiation component is defined, false otherwise.

tfamily \void set(<*type(treeNode)>thisNode) Set the radiation components in the radiation object.

tfamily \doublezero temperature(<integer(:)> [radiationType]→) Return the temperature (in units of Kelvin) of the given radiation structure.

tfamily \doublezero time() The cosmic time at which this radiation object was set.

tfamily regEx

tfamily \void destroy() Destroy the regex.

tfamily \logicalzero matches(<character(len=*>) string→) Return true if a regular expression matches the supplied string.

tfamily rootFinder

tfamily \doublezero find(<double> [rootGuess]|<double(2)> [rootRange]) Find the root of the function given an initial guess or range.

tfamily \logicalzero isInitialized() Return the initialization state of a rootFinder object.

tfamily \void rangeExpand(<double> [rangeExpandUpward]→,<double> [rangeExpandDownward]→, <rangeExpand> [rangeExpandType]→, <double> [rangeUpwardLimit]→, <double> [rangeDownwardLimit]→, <rangeExpandSignExpect> [rangeExpandDownwardSignExpect]→, <rangeExpandSignExpect> [rangeExpandUpwardSignExpect]→) Specify how the initial range will be expanded in a rootFinder object to bracket the root.

tfamily \void rootFunction(<function(<double> x→) rootFunction) Set the function that evaluates \(f(x)\) to use in a rootFinder object.

tfamily \void tolerance(<double> [toleranceAbsolute]→, <double> [toleranceRelative]→) Set the tolerance to use in a rootFinder object.

tfamily \void type(<type(fgsl_root_fsolver_type>) solverType→) Set the type of algorithm to use in a rootFinder object.
tfamily satelliteMergingTimescalesBoylanKolchin2008

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile →) Restore the state of the object from file.

tfamily void stateSnapshot() Stores a snapshot of the object state.

tfamily void stateStore(<integer> stateFile →) Store the state of the object to file.

tfamily double precision timeUntilMerging() Return the time (in Gyr) until the satellite will merge with its host given the current orbit.

tfamily satelliteMergingTimescalesClass

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile →) Restore the state of the object from file.

tfamily void stateSnapshot() Stores a snapshot of the object state.

tfamily void stateStore(<integer> stateFile →) Store the state of the object to file.

tfamily double precision timeUntilMerging() Return the time (in Gyr) until the satellite will merge with its host given the current orbit.

tfamily satelliteMergingTimescalesInfinite

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile →) Restore the state of the object from file.

tfamily void stateSnapshot() Stores a snapshot of the object state.

tfamily void stateStore(<integer> stateFile →) Store the state of the object to file.

tfamily double precision timeUntilMerging() Return the time (in Gyr) until the satellite will merge with its host given the current orbit.

tfamily satelliteMergingTimescalesJiang2008

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile →) Restore the state of the object from file.

tfamily void stateSnapshot() Stores a snapshot of the object state.

tfamily void stateStore(<integer> stateFile →) Store the state of the object to file.

tfamily double precision timeUntilMerging() Return the time (in Gyr) until the satellite will merge with its host given the current orbit.
16.5. Objects

tfamily satelliteMergingTimescalesLaceyCole1993

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile →) Restore the state of the object from file.

tfamily void stateSnapshot() Stores a snapshot of the object state.

tfamily void stateStore(<integer> stateFile →) Store the state of the object to file.

tfamily double precision timeUntilMerging() Return the time (in Gyr) until the satellite will merge with its host given the current orbit.

tfamily double precision timeUntilMergingMassDependence() Return the mass-dependent part of the time (in Gyr) until the satellite will merge with its host.

tfamily satelliteMergingTimescalesLaceyCole1993Tormen

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile →) Restore the state of the object from file.

tfamily void stateSnapshot() Stores a snapshot of the object state.

tfamily void stateStore(<integer> stateFile →) Store the state of the object to file.

tfamily double precision timeUntilMerging() Return the time (in Gyr) until the satellite will merge with its host given the current orbit.

tfamily double precision timeUntilMergingMassDependence() Return the mass-dependent part of the time (in Gyr) until the satellite will merge with its host.

tfamily satelliteMergingTimescalesNull

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile →) Restore the state of the object from file.

tfamily void stateSnapshot() Stores a snapshot of the object state.

tfamily void stateStore(<integer> stateFile →) Store the state of the object to file.

tfamily double precision timeUntilMerging() Return the time (in Gyr) until the satellite will merge with its host given the current orbit.

tfamily satelliteMergingTimescalesPreset

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile →) Restore the state of the object from file.

tfamily void stateSnapshot() Stores a snapshot of the object state.

tfamily void stateStore(<integer> stateFile →) Store the state of the object to file.

tfamily double precision timeUntilMerging() Return the time (in Gyr) until the satellite will merge with its host given the current orbit.
16. Coding GALACTICUS

tfamily satelliteMergingTimescalesVillalobos2013

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile →) Restore the state of the object from file.

tfamily void stateSnapshot() Stores a snapshot of the object state.

tfamily void stateStore(<integer> stateFile →) Store the state of the object to file.

tfamily double precision timeUntilMerging() Return the time (in Gyr) until the satellite will merge with its host given the current orbit.

tfamily satelliteMergingTimescalesWetzelWhite2010

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile →) Restore the state of the object from file.

tfamily void stateSnapshot() Stores a snapshot of the object state.

tfamily void stateStore(<integer> stateFile →) Store the state of the object to file.

tfamily double precision timeUntilMerging() Return the time (in Gyr) until the satellite will merge with its host given the current orbit.

tfamily semaphore

tfamily \void close() Close the semaphore.

tfamily \void post() Post to (i.e. release) a semaphore.

tfamily \void unlink() Unlink a semaphore.

tfamily \void wait() Wait for a semaphore to become available.

tfamily spectraPostprocessorClass

tfamily void apply(<double precision> wavelength→,<double precision> age→,<double precision> redshift→,<double precision> modifier↔) Apply postprocessing to a spectrum.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily spectraPostprocessorIdentity

tfamily void apply(<double precision> wavelength→,<double precision> age→,<double precision> redshift→,<double precision> modifier↔) Apply postprocessing to a spectrum.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily spectraPostprocessorInoue2014

tfamily void apply(<double precision> wavelength→,<double precision> age→,<double precision> redshift→,<double precision> modifier↔) Apply postprocessing to a spectrum.

tfamily logical isFinalizable() Return true if this object can be finalized.
16.5. Objects

tfamily spectraPostprocessorLycSuppress
tfamily void apply(<double precision> wavelength→, <double precision> age→, <double precision> redshift→, <double precision> modifier↔) Apply postprocessing to a spectrum.
tfamily logical isFinalizable() Return true if this object can be finalized..

tfamily spectraPostprocessorMadau1995
tfamily void apply(<double precision> wavelength→, <double precision> age→, <double precision> redshift→, <double precision> modifier↔) Apply postprocessing to a spectrum.
tfamily logical isFinalizable() Return true if this object can be finalized..

tfamily spectraPostprocessorMeiksin2006
tfamily void apply(<double precision> wavelength→, <double precision> age→, <double precision> redshift→, <double precision> modifier↔) Apply postprocessing to a spectrum.
tfamily logical isFinalizable() Return true if this object can be finalized..

tfamily spectraPostprocessorRecent
tfamily void apply(<double precision> wavelength→, <double precision> age→, <double precision> redshift→, <double precision> modifier↔) Apply postprocessing to a spectrum.
tfamily logical isFinalizable() Return true if this object can be finalized..

tfamily stellarLuminosities
tfamily \textcolor{red}{\textless} StellarLuminosities\textgreater add(<type(stellarLuminosities)> stellarLuminosities2→) Add two stellarLuminosities.
tfamily \void builder(<*type(node)> stellarLuminositiesDefinition→) Build a stellar luminosities object from a provided XML description.
tfamily \void deserialize(<double(:)> array→) Deserialize a stellar luminosities object from an array.
tfamily \void destroy() Destroy a stellar luminosities object.
tfamily \textcolor{red}{\textless} StellarLuminosities\textgreater divide(<double> divisor→) Divide stellar luminosities by a scalar.
tfamily \void dump() Dump a stellar luminosities object.
tfamily \void dumpRaw(<integer> fileHandle→) Dump a stellar luminosities object to binary.
tfamily \void increment(<type(stellarLuminosities)> addStellarLuminosities→) Increment a stellar luminosities object.
tfamily \intzero index(<type(varying_string)> name→) Return the index to a luminosity specified by name.
tfamily \logicalzero isOutput(<integer> index→, <double> time→) Return true if the indexed luminosity is to be output at the given time.
tfamily \logicalzero isZero() Return true if a stellar luminosities object is zero.

tfamily \doublezero luminosity(<integer> index\rightarrow) Return the $i^{th}$ luminosity.


tfamily \intzero luminosityCount() Return the total number of luminosities tracked.


tfamily \intzero luminosityOutputCount(<double> time\rightarrow) Return the number of luminosities to be output at the given time.


tfamily \textcolor{red}{\textlesstype(stellarLuminosities)\textgreater} multiply(<double> multiplier\rightarrow) Multiply stellar luminosities by a scalar.


tfamily \textcolor{red}{\textlesstype(varying\_string)\textgreater} name(<integer> index\rightarrow) Return the name of a luminosity specified by index.


tfamily \void output(<integer> integerProperty\leftrightarrow, <integer> integerBufferCount\leftrightarrow, <integer>(::)> integerBuffer\leftrightarrow, <integer>doubleProperty\leftrightarrow, <integer>doubleBufferCount\leftrightarrow, <double>(::)> doubleBuffer\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow) Store a stellar luminosities object in the output buffers.


tfamily \void outputCount(<integer>integerPropertyCount\leftrightarrow, <integer>doublePropertyCount\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow) Specify the count of a stellar luminosities object for output.


tfamily \void outputNames(<integer>integerProperty\leftrightarrow, <char[*](:)>integerPropertyNames\leftrightarrow, <char[*](:)>integerPropertyComments\leftrightarrow, <double(:)>integerPropertyUnitsSI\leftrightarrow, <integer>doubleProperty\leftrightarrow, <char[*](:)>doublePropertyNames\leftrightarrow, <char[*](:)>doublePropertyComments\leftrightarrow, <double(:)>doublePropertyUnitsSI\leftrightarrow, <double> time\rightarrow, <integer> instance\rightarrow) Specify the names of stellar luminosities object properties for output.


tfamily \void readRaw(<integer> fileHandle\rightarrow) Read a stellar luminosities object from binary.


tfamily \void reset() Reset a stellar luminosities object.


tfamily \void serialize(<double(:)> array\leftarrow) Serialize a stellar luminosities object to an array.


tfamily \intzero serializeCount() Return a count of the number of properties in a serialized stellar luminosities object.


tfamily \void setLuminosities(<double> mass\rightarrow, <integer> imfSelected\rightarrow, <double> currentTime\rightarrow, <\textlesstype(abundances)> fuelAbundances\rightarrow) Set the luminosities using a single stellar population.


tfamily \void setToUnity() Set a stellar luminosities object to unity.


tfamily \textcolor{red}{\textlesstype(stellarLuminosities)\textgreater} subtract(<\textlesstype(stellarLuminosities)> stellarLuminosities2\rightarrow) Subtract one abundance from another.


tfamily table

tfamily \void destroy() Destroy the table.
tfamily table1D

tfamily \ void destroy() Destroy the table.

tfamily \ double zero interpolate(<double> x, <integer> [table]) Interpolate to x in the table\textsuperscript{th} table.

tfamily \ double zero interpolateGradient(<double> x, <integer> [table]) Interpolate the gradient to x in the table\textsuperscript{th} table.

tfamily \ logical zero isMonotonic(<enumeration> [direction Decreasing | direction Increasing], <logical> [allowEqual], <integer> [table]) Return true if the table y-values are monotonic. Optionally, the direction of monotonicity can be specified via the \texttt{direction} argument—by default either direction is allowed. By default consecutive equal values are considered non-monotonic. This behavior can be changed via the optional \texttt{allowEqual} argument. If \texttt{table} is specified then the \texttt{table}\textsuperscript{th} table is used for the y-values, otherwise the first table is used.

tfamily \ void reverse(<type(table>) reversedSelf, <integer> [table]) Reverse the table (i.e. swap x and y components) and return in \texttt{reversedSelf}. If \texttt{table} is specified then the \texttt{table}\textsuperscript{th} table is used for the y-values, otherwise the first table is used.

tfamily \ int zero size() Return the size (i.e. number of x-values) in the table.

tfamily \ double zero x(<integer> i) Return the \(i\)th x-value.

tfamily \ double zero xEffective(<double> x) Return the effective value of x to use in table interpolations.

tfamily \ double one xs(<integer> i) Return an array of all x-values.

tfamily \ double zero y(<integer> i, <integer> [table]) Return the \(i\)th y-value. If \texttt{table} is specified then the \texttt{table}\textsuperscript{th} table is used for the y-values, otherwise the first table is used.

tfamily \ double one ys(<integer> i, <integer> [table]) Return an array of all y-values. If \texttt{table} is specified then the \texttt{table}\textsuperscript{th} table is used for the y-values, otherwise the first table is used.

tfamily table1DGeneric

tfamily \ void create(<double(:)> x, <integer> [tableCount]) Create the object with the specified x values, and with \texttt{tableCount} tables.

tfamily \ void destroy() Destroy the table.

tfamily \ double zero interpolate(<double> x, <integer> [table]) Interpolate to x in the table\textsuperscript{th} table.

tfamily \ double zero interpolateGradient(<double> x, <integer> [table]) Interpolate the gradient to x in the table\textsuperscript{th} table.

tfamily \ logical zero isMonotonic(<enumeration> [direction Decreasing | direction Increasing], <logical> [allowEqual], <integer> [table]) Return true if the table y-values are monotonic. Optionally, the direction of monotonicity can be specified via the \texttt{direction} argument—by default either direction is allowed. By default consecutive equal values are considered non-monotonic. This behavior can be changed via the optional \texttt{allowEqual} argument. If \texttt{table} is specified then the \texttt{table}\textsuperscript{th} table is used for the y-values, otherwise the first table is used.

tfamily \ void populate(<double>|<double(:)> y, <integer> [i], <integer> [table]) Populate the \texttt{table}\textsuperscript{th} table with elements y. If y is a scalar, then the index, i, of the element to set must also be specified.
16. Coding Galacticus

tfamily \void reverse(<type(table>) reversedSelf,<integer> [table]) Reverse the table (i.e. swap x and y components) and return in \textit{reversedSelf}. If \textit{table} is specified then the \textit{table}th table is used for the \textit{y}-values, otherwise the first table is used.

tfamily \intzero size() Return the size (i.e. number of \textit{x}-values) in the table.

tfamily \doublezero x(<integer> i) Return the \textit{i}th \textit{x}-value.

tfamily \doublezero xEffective(<double> x) Return the effective value of \textit{x} to use in table interpolations.

tfamily \doubleone xs(<integer> i) Return an array of all \textit{x}-values.

tfamily \doublezero y(<integer> i,<integer> [table]) Return the \textit{i}th \textit{y}-value. If \textit{table} is specified then the \textit{table}th table is used for the \textit{y}-values, otherwise the first table is used.

tfamily \doubleone ys(<integer> i,<integer> [table]) Return an array of all \textit{y}-values. If \textit{table} is specified then the \textit{table}th table is used for the \textit{y}-values, otherwise the first table is used.

tfamily table1DLinearCSpline

tfamily \void create(<double>xMinimum,<double>xMaximum,<integer>xCount,<integer>[tableCount]) Create the object with \textit{x}-values spanning the range \textit{xMinimum} to \textit{xMaximum} in \textit{xCount} steps, and with \textit{tableCount} tables.

tfamily \void destroy() Destroy the table.

tfamily \doublezero interpolate(<double> x,<integer> [table]) Interpolate to \textit{x} in the \textit{table}th table.

tfamily \doublezero interpolateGradient(<double> x,<integer> [table]) Interpolate the gradient to \textit{x} in the \textit{table}th table.

tfamily \logicalzero isMonotonic(<enumeration>[directionDecreasing|directionIncreasing],<logical>[allowEqual],<integer> [table]) Return true if the table \textit{y}-values are monotonic. Optionally, the direction of monotonicity can be specified via the \textit{direction} argument—by default either direction is allowed. By default consecutive equal values are considered non-monotonic. This behavior can be changed via the optional \textit{allowEqual} argument. If \textit{table} is specified then the \textit{table}th table is used for the \textit{y}-values, otherwise the first table is used.

tfamily \void populate(<double>|<double(:)> y,<integer> [i],<integer> [table]) Populate the \textit{table}th table with elements \textit{y}. If \textit{y} is a scalar, then the index, \textit{i}, of the element to set must also be specified.

tfamily \void reverse(<type(table>) reversedSelf,<integer> [table]) Reverse the table (i.e. swap \textit{x} and \textit{y} components) and return in \textit{reversedSelf}. If \textit{table} is specified then the \textit{table}th table is used for the \textit{y}-values, otherwise the first table is used.

tfamily \intzero size() Return the size (i.e. number of \textit{x}-values) in the table.

tfamily \doublezero x(<integer> i) Return the \textit{i}th \textit{x}-value.

tfamily \doublezero xEffective(<double> x) Return the effective value of \textit{x} to use in table interpolations.

tfamily \doubleone xs(<integer> i) Return an array of all \textit{x}-values.

tfamily \doublezero y(<integer> i,<integer> [table]) Return the \textit{i}th \textit{y}-value. If \textit{table} is specified then the \textit{table}th table is used for the \textit{y}-values, otherwise the first table is used.

tfamily \doubleone ys(<integer> i,<integer> [table]) Return an array of all \textit{y}-values. If \textit{table} is specified then the \textit{table}th table is used for the \textit{y}-values, otherwise the first table is used.
16.5. Objects

tfamily 1DLinearLinear

\texttt{void create(<double> xMinimum, <double> xMaximum, <integer> xCount, <integer> [tableCount])}
Create the object with \textit{x}-values spanning the range \texttt{xMinimum} to \texttt{xMaximum} in \texttt{xCount} steps, and with \texttt{tableCount} tables.

tfamily  \texttt{void destroy()} Destroy the table.

ntfamily \texttt{double zero interpolate(<double> x, <integer> [table])} Interpolate to \texttt{x} in the \texttt{table}th table.

ntfamily \texttt{double zero interpolateGradient(<double> x, <integer> [table])} Interpolate the gradient to \texttt{x} in the \texttt{table}th table.

tfamily \texttt{logical zero isMonotonic(<enumeration> [directionDecreasing|directionIncreasing], <logical> [allowEqual], <integer> [table])} Return true if the table \textit{y}-values are monotonic. Optionally, the direction of monotonicity can be specified via the \texttt{direction} argument—by default either direction is allowed. By default consecutive equal values are considered non-monotonic. This behavior can be changed via the optional \texttt{allowEqual} argument. If \texttt{table} is specified then the \texttt{table}th table is used for the \textit{y}-values, otherwise the first table is used.

ntfamily \texttt{void populate(<double>|<double(:)> y, <integer> [i], <integer> [table])} Populate the \texttt{table}th table with elements \texttt{y}. If \texttt{y} is a scalar, then the index, \texttt{i}, of the element to set must also be specified.

ntfamily \texttt{void reverse(<type(table)> reversedSelf, <integer> [table])} Reverse the table (i.e. swap \textit{x} and \textit{y} components) and return in \texttt{reversedSelf}. If \texttt{table} is specified then the \texttt{table}th table is used for the \textit{y}-values, otherwise the first table is used.

ntfamily \texttt{int zero size()} Return the size (i.e. number of \textit{x}-values) in the table.

ntfamily \texttt{double zero x(<integer> i)} Return the \texttt{i}th \textit{x}-value.

ntfamily \texttt{double zero xEffective( <double> x)} Return the effective value of \texttt{x} to use in table interpolations.

ntfamily \texttt{double one xs(<integer> i)} Return an array of all \textit{x}-values.

ntfamily \texttt{double zero y(<integer> i, <integer> [table])} Return the \texttt{i}th \textit{y}-value. If \texttt{table} is specified then the \texttt{table}th table is used for the \textit{y}-values, otherwise the first table is used.

ntfamily \texttt{double one ys(<integer> i, <integer> [table])} Return an array of all \textit{y}-values. If \texttt{table} is specified then the \texttt{table}th table is used for the \textit{y}-values, otherwise the first table is used.

tfamily 1DLogarithmicCSpline

\texttt{void create(<double> xMinimum, <double> xMaximum, <integer> xCount, <integer> [tableCount])}
Create the object with \textit{x}-values spanning the range \texttt{xMinimum} to \texttt{xMaximum} in \texttt{xCount} steps, and with \texttt{tableCount} tables.

tfamily \texttt{void destroy()} Destroy the table.

ntfamily \texttt{double zero interpolate(<double> x, <integer> [table])} Interpolate to \texttt{x} in the \texttt{table}th table.

ntfamily \texttt{double zero interpolateGradient(<double> x, <integer> [table])} Interpolate the gradient to \texttt{x} in the \texttt{table}th table.
\textbf{16. Coding Galacticus}

tfamily \logicalzero isMonotonic(\texttt{\ang{enumeration}>}, \texttt{\ang{directionDecreasing}|\ang{directionIncreasing}}, \texttt{\ang{logical} \ang{allowEqual}}, \texttt{\ang{integer}> \ang{table}}) Return true if the table \textit{y}-values are monotonic. Optionally, the direction of monotonicity can be specified via the \texttt{direction} argument—by default either direction is allowed. By default consecutive equal values are considered non-monotonic. This behavior can be changed via the optional \texttt{allowEqual} argument. If \texttt{table} is specified then the \textit{table}th table is used for the \textit{y}-values, otherwise the first table is used.

tfamily \void populate(\langle \texttt{double} \rangle | \langle \texttt{double(:)} \rangle \textit{y}, \langle \texttt{integer} \rangle \mathit{i}, \langle \texttt{integer} \rangle \mathit{table}) Populate the \textit{table}th table with elements \textit{y}. If \textit{y} is a scalar, then the index, \textit{i}, of the element to set must also be specified.

tfamily \void reverse(\texttt{\ang{type(table)} > reversedSelf}, \langle \texttt{integer} \rangle \mathit{table}) Reverse the table (i.e. swap \textit{x} and \textit{y} components) and return in \texttt{reversedSelf}. If \texttt{table} is specified then the \textit{table}th table is used for the \textit{y}-values, otherwise the first table is used.

tfamily \intzero size() Return the size (i.e. number of \textit{x}-values) in the table.

tfamily \doublezero \textit{x}(\langle \texttt{integer} \rangle \mathit{i}) Return the \textit{i}th \textit{x}-value.

tfamily \doublezero \textit{xEffective}(\langle \texttt{double} > \textit{x}) Return the effective value of \textit{x} to use in table interpolations.

tfamily \doubleone \textit{xs}(\langle \texttt{integer} \rangle \mathit{i}) Return an array of all \textit{x}-values.

tfamily \doublezero \textit{y}(\langle \texttt{integer} \rangle \mathit{i}, \langle \texttt{integer} \rangle \mathit{table}) Return the \textit{i}th \textit{y}-value. If \texttt{table} is specified then the \textit{table}th table is used for the \textit{y}-values, otherwise the first table is used.

tfamily \doubleone \textit{ys}(\langle \texttt{integer} \rangle \mathit{i}, \langle \texttt{integer} \rangle \mathit{table}) Return an array of all \textit{y}-values. If \texttt{table} is specified then the \textit{table}th table is used for the \textit{y}-values, otherwise the first table is used.

tfamily table1DLogarithmicLinear

tfamily \void create(\langle \texttt{double} > \mathit{xMinimum}, \langle \texttt{double} > \mathit{xMaximum}, \langle \texttt{integer} \rangle \mathit{xCount}, \langle \texttt{integer} \rangle \mathit{tableCount}) Create the object with \textit{x}-values spanning the range \textit{xMinimum} to \textit{xMaximum} in \textit{xCount} steps, and with \textit{tableCount} tables.

tfamily \void destroy() Destroy the table.

tfamily \doublezero \textit{interpolate}(\langle \texttt{double} > \mathit{x}, \langle \texttt{integer} \rangle \mathit{table}) Interpolate to \textit{x} in the \textit{table}th table.

tfamily \doublezero \textit{interpolateGradient}(\langle \texttt{double} > \mathit{x}, \langle \texttt{integer} \rangle \mathit{table}) Interpolate the gradient to \textit{x} in the \textit{table}th table.

tfamily \logicalzero isMonotonic(\texttt{\ang{enumeration}>}, \texttt{\ang{directionDecreasing}|\ang{directionIncreasing}}, \texttt{\ang{logical} \ang{allowEqual}}, \texttt{\ang{integer}> \ang{table}}) Return true if the table \textit{y}-values are monotonic. Optionally, the direction of monotonicity can be specified via the \texttt{direction} argument—by default either direction is allowed. By default consecutive equal values are considered non-monotonic. This behavior can be changed via the optional \texttt{allowEqual} argument. If \texttt{table} is specified then the \textit{table}th table is used for the \textit{y}-values, otherwise the first table is used.

tfamily \void populate(\langle \texttt{double} \rangle | \langle \texttt{double(:)} \rangle \textit{y}, \langle \texttt{integer} \rangle \mathit{i}, \langle \texttt{integer} \rangle \mathit{table}) Populate the \textit{table}th table with elements \textit{y}. If \textit{y} is a scalar, then the index, \textit{i}, of the element to set must also be specified.

tfamily \void reverse(\texttt{\ang{type(table)} > reversedSelf}, \langle \texttt{integer} \rangle \mathit{table}) Reverse the table (i.e. swap \textit{x} and \textit{y} components) and return in \texttt{reversedSelf}. If \texttt{table} is specified then the \textit{table}th table is used for the \textit{y}-values, otherwise the first table is used.


\textfamily \intzero size() Return the size (i.e., number of x-values) in the table.

\textfamily \doublezero x(<integer> i) Return the i\textsuperscript{th} x-value.

\textfamily \doublezero xEffective(<double> x) Return the effective value of \(x\) to use in table interpolations.

\textfamily \doubleone xs(<integer> i) Return an array of all x-values.

\textfamily \doublezero y(<integer> i,<integer> [table]) Return the i\textsuperscript{th} y-value. If \texttt{table} is specified then the \(i\textsuperscript{th}\) table is used for the y-values, otherwise the first table is used.

\textfamily \doubleone ys(<integer> i,<integer> [table]) Return an array of all y-values. If \texttt{table} is specified then the \(i\textsuperscript{th}\) table is used for the y-values, otherwise the first table is used.

\textfamily tensorRank2Dimension3Symmetric

\textfamily \textcolor{red}{\textless type(tensorRank2Dimension3Symmetric)\textgreater}add(<type(tensorRank2Dimension3Symmetric)> tensorRank2Dimension3Symmetric2 \rightarrow) Add two tensors.

\textfamily \void builder(<\*type(node)> tensorRank2Dimension3SymmetricDefinition \rightarrow) Build tensor object from a provided XML description.

\textfamily \doublezero contract() Contract a tensor, returning \(T_{ij}\).

\textfamily \void deserialize(<double(:)> tensorArray \rightarrow) Deserialize the tensor object from an array.

\textfamily \void destroy() Destroy a tensor object.

\textfamily \textcolor{red}{\textless type(tensorRank2Dimension3Symmetric)\textgreater}divide(<double> divisor \rightarrow) Divide a tensor by a scalar.

\textfamily \doublezero doubleContract(<type(tensorRank2Dimension3Symmetric)> tensor1) Return the double contraction of two tensors, \(A_{ij}B_{ij}\).

\textfamily \void dump() Dump the tensor object.

\textfamily \void dumpRaw(<integer> fileHandle \rightarrow) Dump the tensor object to binary.

\textfamily \logicalzero equality(<type(tensorRank2Dimension3Symmetric)> self \rightarrow, <double(3,3)> matrix \rightarrow) Return true if a tensor and a matrix are equal.

\textfamily \textcolor{red}{\textless type(tensorRank2Dimension3Symmetric)\textgreater}fromMatrix(double(3,3) matrix) Construct a matrix from a tensor object.

\textfamily \void increment(<type(tensorRank2Dimension3Symmetric)> addTensor \rightarrow) Increment the tensor object.

\textfamily \logicalzero isZero() Return true if a tensor object is zero.

\textfamily \textcolor{red}{\textless type(tensorRank2Dimension3Symmetric)\textgreater}multiply(<double> multiplier \rightarrow) Multiply a tensor by a scalar.

\textfamily \void readRaw(<integer> fileHandle \rightarrow) Read the tensor object from binary.

\textfamily \void reset() Reset elements to zero.

\textfamily \void serialize(<double(:)> tensorArray \leftarrow) Serialize the tensor object to an array.
tfamily \intzero serializeCount() Return a count of the number of properties in a serialized tensor object.

tfamily \void setToIdentity() Set a tensor object to the identity (i.e. all diagonal elements 1, all other elements 0).

tfamily \void setToUnity() Set all elements of the tensor object to unity.

tfamily \textcolor{red}{\textless*class(tensorRank2Dimension3Symmetric)\textgreater} subtract(<type(tensorRank2Dimension3Symmetric)> tensor1, <type(tensorRank2Dimension3Symmetric)> tensor2) Subtract one tensor from another.

tfamily \textcolor{red}{\textless*class(nodeComponentAgeStatistics)\textgreater} ageStatistics(<integer> [instance]→, <logical> [autoCreate]→) Return a ageStatistics component member of the node. If no instance is specified, return the first instance. If autoCreate is true then create a single instance of the component if none exists in the node.

tfamily \intzero ageStatisticsCount() Returns the number of ageStatistics components in the node.

tfamily \void ageStatisticsCreate(<class(nodeComponentAgeStatistics)> [template]→) Create a ageStatistics component in the node. If no template is specified use the active implementation of this class.

tfamily \void ageStatisticsDestroy() Destroy the ageStatistics component(s) of the node.

tfamily \void ageStatisticsMove(<type(treeNode)> targetNode ↔) HASH(0x154d348)

tfamily \void ageStatisticsRemove(<integer> [instance]→) Remove an instance of the ageStatistics component, shifting other instances to keep the array contiguous. If no instance is specified, the first instance is assumed.

tfamily \textcolor{red}{\textless*class(nodeComponentBasic)\textgreater} basic(<integer> [instance]→, <logical> [autoCreate]→) Return a basic component member of the node. If no instance is specified, return the first instance. If autoCreate is true then create a single instance of the component if none exists in the node.

tfamily \intzero basicCount() Returns the number of basic components in the node.

tfamily \void basicCreate(<class(nodeComponentBasic)> [template]→) Create a basic component in the node. If no template is specified use the active implementation of this class.

tfamily \void basicDestroy() Destroy the basic component(s) of the node.

tfamily \void basicMove(<type(treeNode)> targetNode ↔) HASH(0x152eb70)

tfamily \void basicRemove(<integer> [instance]→) Remove an instance of the basic component, shifting other instances to keep the array contiguous. If no instance is specified, the first instance is assumed.

tfamily \textcolor{red}{\textless*class(nodeComponentBlackHole)\textgreater} blackHole(<integer> [instance]→, <logical> [autoCreate]→) Return a blackHole component member of the node. If no instance is specified, return the first instance. If autoCreate is true then create a single instance of the component if none exists in the node.
16.5. Objects

tfamily \intzero blackHoleCount() Returns the number of blackHole components in the node.
tfamily \void blackHoleCreate(<class(nodeComponentBlackHole)> [template]→) Create a blackHole component in the node. If no template is specified use the active implementation of this class.
tfamily \void blackHoleDestroy() Destroy the blackHole component(s) of the node.
tfamily \void blackHoleMove(<type(treeNode)> targetNode↔) HASH(0x1552808)
tfamily \void blackHoleRemove(<integer> [instance]→) Remove an instance of the blackHole component, shifting other instances to keep the array contiguous. If no instance is specified, the first instance is assumed.
tfamily \void componentBuilder(<*type(node)> nodeDefinition→) Build components in this node given an XML description of their properties.
tfamily \void copyNodeTo(<class(treeNode)> targetNode↔, <logical> [skipFormationNode]→) Make a copy of the node in targetNode. If skipFormationNode is true then do not copy any pointer to the formation node.
tfamily \textcolor{red}{\textless *type(nodeEvent)\textgreater} createEvent() Create a nodeEvent object in this node.
tfamily \textcolor{red}{\textless *class(nodeComponentDarkMatterProfile)\textgreater} darkMatterProfile(<integer> [instance]→, <logical> [autoCreate]→) Return a darkMatterProfile component member of the node. If no instance is specified, return the first instance. If autoCreate is true then create a single instance of the component if none exists in the node.
tfamily \intzero darkMatterProfileCount() Returns the number of darkMatterProfile components in the node.
tfamily \void darkMatterProfileCreate(<class(nodeComponentDarkMatterProfile)> [template]→) Create a darkMatterProfile component in the node. If no template is specified use the active implementation of this class.
tfamily \void darkMatterProfileDestroy() Destroy the darkMatterProfile component(s) of the node.
tfamily \void darkMatterProfileMove(<type(treeNode)> targetNode↔) HASH(0x1556478)
tfamily \void darkMatterProfileRemove(<integer> [instance]→) Remove an instance of the darkMatterProfile component, shifting other instances to keep the array contiguous. If no instance is specified, the first instance is assumed.
tfamily \void deserializeRates(<double(:)> array→) Deserialize rates from array.
tfamily \void deserializeScales(<double(:)> array→) Deserialize scales from array.
tfamily \void deserializeValues(<double(:)> array→) Deserialize values from array.
tfamily \void destroy() Destroy this node.
tfamily \textcolor{red}{\textless *class(nodeComponentDisk)\textgreater} disk(<integer> [instance]→, <logical> [autoCreate]→) Return a disk component member of the node. If no instance is specified, return the first instance. If autoCreate is true then create a single instance of the component if none exists in the node.
tfamily `\textcolor{red}\textless*\text{intzero}\textgreater\text{diskCount()}` Returns the number of `disk` components in the node.

tfamily `\textcolor{red}\textless*\text{void diskCreate(<class(nodeComponentDisk)> [template]→)}` Create a `disk` component in the node. If no `template` is specified use the active implementation of this class.

tfamily `\textcolor{red}\textless*\text{void diskDestroy()}` Destroy the `disk` component(s) of the node.

tfamily `\textcolor{red}\textless*\text{void diskMove(<type(treeNode)> targetNode↔) HASH(0x15abd98)}`

tfamily `\textcolor{red}\textless*\text{void diskRemove(<integer> [instance]→)}` Remove an instance of the disk component, shifting other instances to keep the array contiguous. If no `instance` is specified, the first instance is assumed.

tfamily `\textcolor{red}\textless*\text{void dump()}` Generate an ASCII dump of all content of a node.

tfamily `\textcolor{red}\textless*\text{void dumpRaw(<integer> fileHandle→)}` Generate a binary dump of all content of a node.

tfamily `\textcolor{red}\textless*\text{void dumpXML()}` Generate an XML dump of all content of a node.

tfamily `\textcolor{red}\textless*\text{intzero dynamicsStatisticsCount()}` Returns the number of `dynamicsStatistics` components in the node.

tfamily `\textcolor{red}\textless*\text{void dynamicsStatisticsCreate(<class(nodeComponentDynamicsStatistics)> [template]→)}` Create a `dynamicsStatistics` component in the node. If no `template` is specified use the active implementation of this class.

tfamily `\textcolor{red}\textless*\text{void dynamicsStatisticsDestroy()}` Destroy the `dynamicsStatistics` component(s) of the node.

tfamily `\textcolor{red}\textless*\text{void dynamicsStatisticsMove(<type(treeNode)> targetNode↔) HASH(0x154d030)}`

tfamily `\textcolor{red}\textless*\text{void dynamicsStatisticsRemove(<integer> [instance]→)}` Remove an instance of the dynamicsStatistics component, shifting other instances to keep the array contiguous. If no `instance` is specified, the first instance is assumed.

tfamily `\textcolor{red}\textless*\text{type(treeNode)	extgreater earliestProgenitor()}` Return a pointer to the earliest progenitor (along the main branch) of this node.

tfamily `\textcolor{red}\textless*\text{class(nodeComponentFormationTime)	extgreater formationTime(<integer> [instance]→, <logical> [autoCreate]→)}` Return a formationTime component member of the node. If no `instance` is specified, return the first instance. If `autoCreate` is `true` then create a single instance of the component if none exists in the node.

tfamily `\textcolor{red}\textless*\text{intzero formationTimeCount()}` Returns the number of `formationTime` components in the node.

tfamily `\textcolor{red}\textless*\text{void formationTimeCreate(<class(nodeComponentFormationTime)> [template]→)}` Create a `formationTime` component in the node. If no `template` is specified use the active implementation of this class.

tfamily `\textcolor{red}\textless*\text{void formationTimeDestroy()}` Destroy the `formationTime` component(s) of the node.

tfamily `\textcolor{red}\textless*\text{void formationTimeMove(<type(treeNode)> targetNode↔) HASH(0x15527f0)}`
16.5. Objects

tfamily \void formationTimeRemove(<integer> [instance] \rightarrow) Remove an instance of the formationTime component, shifting other instances to keep the array contiguous. If no instance is specified, the first instance is assumed.

tfamily \textcolor{red}{\textless*class(nodeComponentHostHistory)\textgreater} hostHistory(<integer> [instance] \rightarrow, <logical> [autoCreate] \rightarrow) Return a hostHistory component member of the node. If no instance is specified, return the first instance. If autoCreate is true then create a single instance of the component if none exists in the node.

tfamily \intzero hostHistoryCount() Returns the number of hostHistory components in the node.

tfamily \void hostHistoryCreate(<class(nodeComponentHostHistory)> [template] \rightarrow) Create a hostHistory component in the node. If no template is specified use the active implementation of this class.

tfamily \void hostHistoryDestroy() Destroy the hostHistory component(s) of the node.

tfamily \void hostHistoryMove(<type(treeNode)> targetNode \leftrightarrow) HASH(0x1558a18)

tfamily \void hostHistoryRemove(<integer> [instance] \rightarrow) Remove an instance of the hostHistory component, shifting other instances to keep the array contiguous. If no instance is specified, the first instance is assumed.

tfamily \textcolor{red}{\textless*class(nodeComponentHotHalo)\textgreater} hotHalo(<integer> [instance] \rightarrow, <logical> [autoCreate] \rightarrow) Return a hotHalo component member of the node. If no instance is specified, return the first instance. If autoCreate is true then create a single instance of the component if none exists in the node.

tfamily \intzero hotHaloCount() Returns the number of hotHalo components in the node.

tfamily \void hotHaloCreate(<class(nodeComponentHotHalo)> [template] \rightarrow) Create a hotHalo component in the node. If no template is specified use the active implementation of this class.

tfamily \void hotHaloDestroy() Destroy the hotHalo component(s) of the node.

tfamily \void hotHaloMove(<type(treeNode)> targetNode \leftrightarrow) HASH(0x150af68)

tfamily \void hotHaloRemove(<integer> [instance] \rightarrow) Remove an instance of the hotHalo component, shifting other instances to keep the array contiguous. If no instance is specified, the first instance is assumed.

tfamily \textcolor{red}{\textless integer(kind\_int8)\textgreater} index() Return the index of this node.

tfamily \void indexSet(<integer(kind\_int8)> index \rightarrow) Set the index of this node.

tfamily \textcolor{red}{\textless*class(nodeComponentIndices)\textgreater} indices(<integer> [instance] \rightarrow, <logical> [autoCreate] \rightarrow) Return a indices component member of the node. If no instance is specified, return the first instance. If autoCreate is true then create a single instance of the component if none exists in the node.

tfamily \intzero indicesCount() Returns the number of indices components in the node.

tfamily \void indicesCreate(<class(nodeComponentIndices)> [template] \rightarrow) Create a indices component in the node. If no template is specified use the active implementation of this class.

tfamily \void indicesDestroy() Destroy the indices component(s) of the node.
tfamily \void indicesMove(<\text{type}(\text{treeNode})> \text{targetNode} \leftrightarrow) \text{HASH}(0x1550198)

tfamily \void indicesRemove(<\text{integer}> [\text{instance}]\rightarrow) \text{Remove an instance of the indices component, shifting other instances to keep the array contiguous. If no \text{instance} is specified, the first instance is assumed.}

tfamily \void initialize(<\text{integer}(\text{kind_int8})> \text{index} \rightarrow) \text{Initialize this node (assigns a unique identifier, creates generic components).}

tfamily \textcolor{red}{\langle*\text{class(nodeComponentInterOutput)}\rangle} \text{interOutput}(<\text{integer}> \text{[instance]}\rightarrow, <\text{logical}> [\text{autoCreate}]\rightarrow) \text{Return a interOutput component member of the node. If no \text{instance} is specified, return the first instance. If \text{autoCreate} is \text{true} then create a single instance of the component if none exists in the node.}

tfamily \text{intzero} \text{interOutputCount}() \text{Returns the number of \text{interOutput} components in the node.}

tfamily \void \text{interOutputCreate}(<\text{class(nodeComponentInterOutput)}>[\text{template}]\rightarrow) \text{Create a \text{interOutput} component in the node. If no \text{template} is specified use the active implementation of this class.}

tfamily \void \text{interOutputDestroy}() \text{Destroy the \text{interOutput} component(s) of the node.}

tfamily \void \text{interOutputMove}(<\text{type}(\text{treeNode})> \text{targetNode} \leftrightarrow) \text{HASH}(0x15581a8)

tfamily \void \text{interOutputRemove}(<\text{integer}> [\text{instance}]\rightarrow) \text{Remove an instance of the \text{interOutput} component, shifting other instances to keep the array contiguous. If no \text{instance} is specified, the first instance is assumed.}

tfamily \logicalzero \text{isOnMainBranch}() \text{Return true if this node is on the main branch of its tree, false otherwise.}

tfamily \logicalzero \text{isPrimaryProgenitor}() \text{Return true if this node is the primary progenitor of its descendent, false otherwise.}

tfamily \logicalzero \text{isPrimaryProgenitorOf(<\text{integer}(\text{kind_int8})> \text{targetNodeIndex} \rightarrow|<*\text{type}(\text{treeNode})> \text{targetNode} \rightarrow)} \text{Return true is this node is the primary progenitor of the specified (by index or pointer) node, false otherwise.}

tfamily \logicalzero \text{isProgenitorOf(<\text{integer}(\text{kind_int8})> \text{targetNodeIndex} \rightarrow|<*\text{type}(\text{treeNode})> \text{targetNode} \rightarrow)} \text{Return true is this node is a progenitor of the specified (by index or pointer) node, false otherwise.}

tfamily \logicalzero \text{isSatellite}() \text{Return true if this node is a satellite, false otherwise.}

tfamily \textcolor{red}{\langle*\text{type}(\text{treeNode})\rangle} \text{lastSatellite}() \text{Return a pointer to the last satellite in the list of satellites belonging to this node.}

tfamily \text{doublezero} mapDouble0(<*\text{function()}> \text{mapFunction}) \text{Map a scalar double function over components.}

tfamily \void mapVoid(<*\text{function()}> \text{mapFunction}) \text{Map a void function over components.}

tfamily \textcolor{red}{\langle*\text{class(nodeComponentMassFlowStatistics)}\rangle} \text{massFlowStatistics}(<\text{integer}> [\text{instance}]\rightarrow, <\text{logical}> [\text{autoCreate}]\rightarrow) \text{Return a massFlowStatistics component member of the node. If no \text{instance} is specified, return the first instance. If \text{autoCreate} is \text{true} then create a single instance of the component if none exists in the node.}
16.5. Objects

tfamily \texttt{\textcolor{red}{\textless *type(treeNode)\textgreater} mergeWith()} Return a pointer to the node with which this node will merge.

tfamily \texttt{\textcolor{red}{\textless *class(nodeComponentMergingStatistics)\textgreater} mergingStatistics(<integer>[instance]\rightarrow, <logical>[autoCreate]\rightarrow)} Return a mergingStatistics component member of the node. If no instance is specified, return the first instance. If autoCreate is true then create a single instance of the component if none exists in the node.

tfamily \texttt{\textcolor{red}{\textless varying\_string\textgreater} nameFromIndex(<integer>index\rightarrow)} Return the name of a property given its index in a node.

tfamily \texttt{\textcolor{red}{\textless*class(nodeComponentNBody)\textgreater} nBody(<integer>[instance]\rightarrow, <logical>[autoCreate]\rightarrow)} Return a nBody component member of the node. If no instance is specified, return the first instance. If autoCreate is true then create a single instance of the component if none exists in the node.

tfamily \texttt{\textcolor{red}{\textless varying\_string\textgreater} nameFromIndex(<integer> index\rightarrow)} Return the name of a property given its index in a node.

tfamily \texttt{\textcolor{red}{\textless *type(treeNode)\textgreater} mergeWith()} Return a pointer to the node with which this node will merge.
tfamily \void nBodyMove(<\text{type}\langle\text{treeNode}\rangle> targetNode\leftarrow) HASH(0x1558ec8)

tfamily \void nBodyRemove(<\text{integer}> [\text{instance}]\rightarrow) Remove an instance of the nBody component, shifting other instances to keep the array contiguous. If no \text{instance} is specified, the first instance is assumed.

tfamily \void odeStepRatesInitialize() Initialize rates of evolvable properties.

tfamily \void odeStepScalesInitialize() Initialize tolerance scales of evolvable properties.

tfamily \void output(<\text{integer}> integerProperty\leftrightarrow, <\text{integer}> integerBufferCount\leftrightarrow, <\text{integer}\langle:\rangle> integerBuffer\leftrightarrow, <\text{integer}> doubleProperty\leftrightarrow, <\text{integer}> doubleBufferCount\leftrightarrow, <\text{double}\langle:\rangle> doubleBuffer\leftrightarrow, <\text{double}> time\rightarrow) Populate output buffers with properties for a node.

tfamily \void outputCount(<\text{integer}> integerPropertyCount\leftrightarrow, <\text{integer}> doublePropertyCount\leftrightarrow, <\text{double}> time\rightarrow) Increment the count of properties to output for a node.

tfamily \void outputNames(<\text{integer}> integerProperty\leftrightarrow, <\text{char}\langle\text{\*}\rangle\langle:\rangle> integerPropertyNames\leftrightarrow, <\text{char}\langle\text{\*}\rangle\langle:\rangle> integerPropertyComments\leftrightarrow, <\text{double}\langle:\rangle> integerPropertyUnitsSI\leftrightarrow, <\text{integer}> doubleProperty\leftrightarrow, <\text{char}\langle\text{\*}\rangle\langle:\rangle> doublePropertyNames\leftrightarrow, <\text{char}\langle\text{\*}\rangle\langle:\rangle> doublePropertyComments\leftrightarrow, <\text{double}\langle:\rangle> doublePropertyUnitsSI\leftrightarrow, <\text{double}> time\rightarrow) Establish the names of properties to output for a node.

tfamily \\textcolor{red}\langle\text{\*class}(<\text{nodeComponentPosition}\rangle)\textgreater\ texttt{position(<\text{integer}> [\text{instance}]\rightarrow, <\text{logical}> [\text{autoCreate}]\rightarrow)} Return a position component member of the node. If no \text{instance} is specified, return the first instance. If \text{autoCreate} is \texttt{true} then create a single instance of the component if none exists in the node.

tfamily \\texttt{positionCount()} Returns the number of \texttt{position} components in the node.

tfamily \\void positionCreate(<\text{class}\langle\text{nodeComponentPosition}\rangle> [\text{template}]\rightarrow) Create a \text{position} component in the node. If no \texttt{template} is specified use the active implementation of this class.

tfamily \\void positionDestroy() Destroy the \texttt{position} component(s) of the node.

tfamily \\void positionMove(<\text{type}\langle\text{treeNode}\rangle> targetNode\leftarrow) HASH(0x150fe38)

tfamily \\void positionRemove(<\text{integer}> [\text{instance}]\rightarrow) Remove an instance of the position component, shifting other instances to keep the array contiguous. If no \text{instance} is specified, the first instance is assumed.

tfamily \\void readRaw(<\text{integer}> fileHandle\rightarrow) Read a binary dump of all content of a node.

tfamily \\void removeFromHost() Remove this node from the satellite population of its host halo.

tfamily \\void removeFromMergee() Remove this node from the list of mergees associated with its merge target.

tfamily \\void removePairedEvent(<\text{type}\langle\text{nodeEvent}\rangle> event\rightarrow) Remove a paired \text{nodeEvent} from this node.

tfamily \textcolor{red}\langle\text{\*class}\langle\text{nodeComponentSatellite}\rangle\textgreater\ satellite(<\text{integer}> [\text{instance}]\rightarrow, <\text{logical}> [\text{autoCreate}]\rightarrow) Return a satellite component member of the node. If no \text{instance} is specified, return the first instance. If \text{autoCreate} is \texttt{true} then create a single instance of the component if none exists in the node.

tfamily \\texttt{satelliteCount()} Returns the number of \texttt{satellite} components in the node.
16.5. Objects

tfamily \void satelliteCreate(<class(nodeComponentSatellite)> [template]->) Create a satellite component in the node. If no template is specified use the active implementation of this class.

tfamily \void satelliteDestroy() Destroy the satellite component(s) of the node.

tfamily \void satelliteMove(<type(treeNode)> targetNode<->) HASH(0x15504b0)

tfamily \void satelliteRemove(<integer> [instance]->) Remove an instance of the satellite component, shifting other instances to keep the array contiguous. If no instance is specified, the first instance is assumed.

tfamily \intzero serializeCount() Return a count of the number of evolvable properties of the serialized object.

tfamily \void serializeRates(<double(:)> array<->) Serialize rates to array.

tfamily \void serializeScales(<double(:)> array<->) Serialize scales to array.

tfamily \void serializeValues(<double(:)> array<->) Serialize values to array.

tfamily \textcolor{red}{\textless*class(nodeComponentSpheroid)\textgreater}spheroid(<integer>[instance]->, <logical> [autoCreate]->) Return a spheroid component member of the node. If no instance is specified, return the first instance. If autoCreate is true then create a single instance of the component if none exists in the node.

tfamily \intzero spheroidCount() Returns the number of spheroid components in the node.

tfamily \void spheroidCreate(<class(nodeComponentSpheroid)> [template]->) Create a spheroid component in the node. If no template is specified use the active implementation of this class.

tfamily \void spheroidDestroy() Destroy the spheroid component(s) of the node.

tfamily \void spheroidMove(<type(treeNode)> targetNode<->) HASH(0x15526a0)

tfamily \void spheroidRemove(<integer> [instance]->) Remove an instance of the spheroid component, shifting other instances to keep the array contiguous. If no instance is specified, the first instance is assumed.

tfamily \textcolor{red}{\textless*class(nodeComponentSpin)\textgreater}spin(<integer>[instance]->, <logical> [autoCreate]->) Return a spin component member of the node. If no instance is specified, return the first instance. If autoCreate is true then create a single instance of the component if none exists in the node.

tfamily \intzero spinCount() Returns the number of spin components in the node.

tfamily \void spinCreate(<class(nodeComponentSpin)> [template]->) Create a spin component in the node. If no template is specified use the active implementation of this class.

tfamily \void spinDestroy() Destroy the spin component(s) of the node.

tfamily \void spinMove(<type(treeNode)> targetNode<->) HASH(0x1558370)

tfamily \void spinRemove(<integer> [instance]->) Remove an instance of the spin component, shifting other instances to keep the array contiguous. If no instance is specified, the first instance is assumed.
16. Coding GALACTICUS

tfamily \textcolor{red}{\textless \textless type(varying\_string)\textgreater} type() Return the type of this node.

tfamily \textcolor{red}{\textless integer(kind\_int8)\textgreater} uniqueID() Return the unique identifier for this node.

tfamily \textcolor{red}{\textless integer(kind\_int8)\textgreater} uniqueIDSet(<integer(kind_int8>) uniqueID\rightarrow) Set the unique identifier for this node.

tfamily \void walkBranch(<type(treeNode)> startNode\leftrightarrow, <type(treeNode)> nextNode\leftrightarrow) Return a pointer to the next node when performing a walk of a single branch of the tree, excluding satellites.

tfamily \void walkBranchWithSatellites(<type(treeNode)> startNode\leftrightarrow, <type(treeNode)> nextNode\leftrightarrow) Return a pointer to the next node when performing a walk of a single branch of the tree, including satellites.

tfamily \void walkTree(<type(treeNode)> nextNode\leftrightarrow) Return a pointer to the next node when performing a walk of the entire tree, excluding satellites.

tfamily \void walkTreeUnderConstruction(<type(treeNode)> nextNode\leftrightarrow) Return a pointer to the next node when performing a walk of a tree under construction.

tfamily \void walkTreeWithSatellites(<type(treeNode)> nextNode\leftrightarrow) Return a pointer to the next node when performing a walk of the entire tree, including satellites.

tfamily universe

tfamily \textcolor{red}{\textless *type(universeEvent)\textgreater} createEvent() Create a treeEvent object in this universe.

tfamily \textcolor{red}{\textless *type(mergerTree)\textgreater} popTree() Pop a mergerTree from this universe.

tfamily \void pushTree(<type(mergerTree)> thisTree\rightarrow) Pop a mergerTree from this universe.

tfamily \void removeEvent(<type(universeEvent)> event\rightarrow) Remove a treeEvent from this universe.

tfamily virialDensityContrastBryanNorman1998

tfamily double precision densityContrast(<double precision> time\rightarrow, <double precision> expansionFactor\rightarrow, <logical> collapsing\rightarrow) Returns the virial density contrast at the given epoch.

tfamily double precision densityContrastRateOfChange(<double precision> time\rightarrow, <double precision> expansionFactor\rightarrow, <logical> collapsing\rightarrow) Returns the rate of change of virial density contrast at the given epoch.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile\rightarrow) Restore the state of the object to file.

tfamily void stateStore(<integer> stateFile\rightarrow) Store the state of the object to file.
16.5. Objects

tfamily virialDensityContrastClass

tfamily double precision densityContrast(<double precision> time→,<double precision> expansionFactor→,<logical> collapsing→) Returns the virial density contrast at the given epoch.

tfamily double precision densityContrastRateOfChange(<double precision> time→,<double precision> expansionFactor→,<logical> collapsing→) Returns the rate of change of virial density contrast at the given epoch.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile→) Restore the state of the object to file.

tfamily void stateStore(<integer> stateFile→) Store the state of the object to file.

tfamily virialDensityContrastFixed


tfamily double precision densityContrast(<double precision> time→,<double precision> expansionFactor→,<logical> collapsing→) Returns the virial density contrast at the given epoch.

tfamily double precision densityContrastRateOfChange(<double precision> time→,<double precision> expansionFactor→,<logical> collapsing→) Returns the rate of change of virial density contrast at the given epoch.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile→) Restore the state of the object to file.

tfamily void stateStore(<integer> stateFile→) Store the state of the object to file.


tfamily virialDensityContrastFriendsOfFriends


tfamily double precision densityContrast(<double precision> time→,<double precision> expansionFactor→,<logical> collapsing→) Returns the virial density contrast at the given epoch.

tfamily double precision densityContrastRateOfChange(<double precision> time→,<double precision> expansionFactor→,<logical> collapsing→) Returns the rate of change of virial density contrast at the given epoch.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile→) Restore the state of the object to file.

tfamily void stateStore(<integer> stateFile→) Store the state of the object to file.


tfamily virialDensityContrastKitayamaSuto1996


tfamily double precision densityContrast(<double precision> time→,<double precision> expansionFactor→,<logical> collapsing→) Returns the virial density contrast at the given epoch.

tfamily double precision densityContrastRateOfChange(<double precision> time→,<double precision> expansionFactor→,<logical> collapsing→) Returns the rate of change of virial density contrast at the given epoch.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void stateRestore(<integer> stateFile→) Restore the state of the object to file.

tfamily void stateStore(<integer> stateFile→) Store the state of the object to file.
tfamily virialDensityContrastSphericalCollapseMatterDE

tfamily double precision densityContrast(<double precision> time→,<double precision> expansionFactor→,<logical> collapsing→) Returns the virial density contrast at the given epoch.

tfamily double precision densityContrastRateOfChange(<double precision> time→,<double precision> expansionFactor→,<logical> collapsing→) Returns the rate of change of virial density contrast at the given epoch.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void retabulate(<double> time→) Tabulate spherical collapse virial density contrast.

tfamily void stateRestore(<integer> stateFile→) Restore the state of the object to file.

tfamily void stateStore(<integer> stateFile→) Store the state of the object to file.

tfamily virialDensityContrastSphericalCollapseMatterLambda

tfamily double precision densityContrast(<double precision> time→,<double precision> expansionFactor→,<logical> collapsing→) Returns the virial density contrast at the given epoch.

tfamily double precision densityContrastRateOfChange(<double precision> time→,<double precision> expansionFactor→,<logical> collapsing→) Returns the rate of change of virial density contrast at the given epoch.

tfamily logical isFinalizable() Return true if this object can be finalized.

tfamily void retabulate(<double> time→) Tabulate spherical collapse virial density contrast.

tfamily void stateRestore(<integer> stateFile→) Restore the state of the object to file.

tfamily void stateStore(<integer> stateFile→) Store the state of the object to file.
17. Adding New Methods

17.1. Code Directives

GALACTICUS is designed to be flexible and extensible, allowing you to add new methods and functionality without having to hack the code extensively. To achieve this it makes much use of embedded code directives which, for example, explain to the build system how a particular subroutine or function connects into the GALACTICUS code. Such code directives are indicated by lines beginning with !#, and take the form of short blocks of XML. For example, a typical code directive might look like:

```xml
!# <accretionDisksMethod>
!# <unitName>Accretion_Disks_Shakura_Sunyaev_Initialize</unitName>
!# </accretionDisksMethod>
```

This directive would typically appear just prior to a subroutine which initializes the Shakura-Sunyaev accretion disk module (it could appear anywhere throughout that module, but it makes sense to keep it close to the subroutine that it references). The accretionDisksMethod tag explains to the GALACTICUS build system that this module contains an implementation of black hole accretion disks. The unitName tag specifies the name of a program unit which (in this case) should be called to initialize this accretion disk implementation. The build system will then insert appropriate use and call statements into the GALACTICUS code such that this routine will be called if and when accretion disks are required by GALACTICUS.

17.2. Identifying Components and Mass Types

Many functions can be applied to different components or groups of components and to different types of mass within a node. In general, these functions make use of a set of label defined in the Galactic_Structure_Options module. Components are identified by a componentType label which can take on the following values:

- **componentTypeAll** All components are matched;
- **componentTypeDisk** Only disk components are matched;
- **componentTypeSpheroid** Only spheroid components are matched.
- **componentTypeBlackHole** Only black hole components are matched.
- **componentTypeHotHalo** Only hot halo components are matched.
- **componentTypeDarkHalo** Only dark matter halo components are matched.

Types of mass are identified by a massType which can take one of the following values:

- **massTypeAll** All mass is included;
- **massTypeDark** Only dark matter is included;
massTypeBaryonic Only baryonic mass is included.
massTypeGalactic Only galactic mass is included.
massTypeGaseous Only gaseous mass is included.
massTypeStellar Only stellar mass is included.
massTypeBlackHole Only black hole mass is included.

17.3. Components

This section describes the internal structure of node components, and how a component is implemented.

17.3.1. Component Structure

Each node in the merger tree consists of an arbitrary number of “components”, each of which can actually be an array, allowing multiple components of a given class. Each component represents a specific class of object, which could be a dark matter halo, a galactic disk or a black hole etc. A component of each class may be of one or more different implementations of that component class. Component classes are extensions of the nodeComponent base class, while each implementation is an extension of its component class (or, sometimes, of another implementation of that same class). Each component implementation type consists of a set of data, representing the properties (mass, size etc.) of the component, along with the rates of change (and ODE solver tolerances) for any properties which are evolvable. Additionally, each component contains a large number of methods (functions) which can be used to access its properties, query its interfaces and which are used internally to perform ODE evolution, output etc. The nodeComponent base class and all classes derived from it are built automatically by Galacticus::Build::Components at compile time (take a look in work/build/objects.nodes.components.Inc if you want to see the generated code).

17.3.2. Extending Components

It is possible to create a component which extends an existing component (see the discussion of the extends element in §17.3.3). This capability is intended to allow new properties to be added to a component without having to create a whole new copy of the component. It is not intended to allow changes in the way in which the component is evolved through the halo hierarchy. (With the exception that rules to describe how the newly added properties will evolve through the halo hierarchy can be added of course.)

A simple example of this extension capability can be found in the scaleShape dark matter profile component (§12.13.3), which extends the scale dark matter profile component (§12.13.1). In this case, the scaleShape component adds a new property, shape, and specifies how it is to be initialized, evolved, output, and change by node promotion events. It does not affect how the scale property, inherited from the scale dark matter profile component, is evolved.

\footnote{Data objects in components can be real, integer, boolean or of derived type. For derived types, currently history, abundances, chemicals, and keplerOrbit are supported. Adding additional derived types is possible, providing that the type supports the required methods for output, serialization, etc. Data objects can currently be scalar or rank-1 arrays.}
17.3. Implementing a New Component

Implementing a new component involves writing some modules and functions which contain a definition of the component and, if necessary, handle initialization, creation, evolution, and responses to any events. Frequently, the easiest way to make a new component is to copy a previously existing one and modify it as needed. Details of the various functions that component modules must perform are given below.

By convention, a component’s implementation is split into three or four files, although some components might not need all of these files. These files are named as follows (with `<component>` acting as a placeholder for the name of the component in question):

- `objects.nodes.components.<component>.F90` The primary file which describes the component and its properties, and which contains functions that manipulate the component as it evolves through a merger tree (ODE rates, behavior during mergers, etc.);
- `objects.nodes.components.<component>.bound_functions.F90` Contains functions which will be bound to the component object (i.e. the `nodeComponent<Class><Implementation>` class), and so will be available as type bound procedures. Generally, these functions will include any which get or set values of properties in the component, those which return information about its internal state (such as the density at some position in the component; see §17.4.3), and any other functions which we may want to be overridden by extensions to the component.
- `objects.nodes.components.<component>.data.F90` Contains any data which may need to be shared between the above two files. This might contain parameters which control some property of the component that is the same for all instances (e.g. if spheroids are modelled as Sérsic profiles all with the same value of the Sérsic index, that value might be placed into this file).
- `objects.nodes.components.<component>.structure.F90` Contains any functions which implement the structure (e.g. density, rotation curve) of the component and which cannot be placed in `objects.nodes.components.<component>.bound_functions.Inc` due to dependencies on modules which in turn depend on the `Galacticus_Nodes` module.

In general, `objects.nodes.components.<component>.F90` is the place for the component definition and functions which process the component during tree evolution (including output), while `objects.nodes.components.<component>.bound_functions.Inc` is intended for functions which record or report the internal state of the component.

Component Definition

Component definition itself takes the form of an embedded XML document. The following example illustrates such a document:

```xml
<component>
  <class>disk</class>
  <name>exponential</name>
  <isDefault>yes</isDefault>
  <properties>
    <property>
      <name>isInitialized</name>
      <type>logical</type>
      <rank>0</rank>
      <attributes isSettable="true" isGettable="true" isEvolvable="false" />
    </property>
    <property>
      <name>massStellar</name>
    </property>
  </properties>
</component>
```
17. Adding New Methods

The elements of this document have the following meaning:
class [Required] Specifies the component class of which this is an implementation.

name [Required] Specifies the name of this specific implementation.

extends [Optional] If present, this element must contain class and name elements which specify the type of component which should be extended. The component then automatically inherits all properties and type-bound functions of the extended type.

isDefault [Required] Specifies whether or not this should be the default implementation of this class. Note that only one implementation of each class can be declared to be the default. If no implementation of a given class is declared to be the default then the (automatically generated) null implementation will be made the default.

properties [Optional] Contains an array of property elements which specify the properties of this implementation. Each member property has the following structure:

name [Required] The name of the property.

type [Required] The type (one of real, integer, logical, history, abundances, chemicals, or keplerOrbit at present) of the property.

rank [Required] The rank of this property (currently 0 for a scalar or 1 for a 1-D array).

attributes [Required] Attributes of this property:

isSettable If true then the value of this property can be set directory.

isGettable If true then the value of this property can be got directory.

isEvolvable If true this property evolves as part of the GALACTICUS ODE system.

createIfNeeded If true then any attempt to get, set, or adjust the rate of this property will cause the component to be created if it does not already exist. This is useful if the component should be created in response to mass transfer from some other component for example.

makeGeneric If true then any rate method for this property will have a version created which binds to the base nodeComponent class. This version is suitable for attaching to deferred rate functions of components of another class. For example, the disk gas mass rate function is made generic, and then attached to the deferred cooling rate of the hot halo using:

call hotHalo%hotHaloCoolingMassRateFunction(DiskExponentialMassGasRateGeneric)

isDeferred Contains a “:” separated list which can contain get, set, and rate. The methods present in this list will not have functions bound to them at compile time. Instead a function will be created which allows a function to be bound to these methods at run time. For example:

call myComponent%massFunction (My_Component_Mass_Get_Function)
call myComponent%massSetFunction (My_Component_Mass_Set_Function)
call myComponent%massRateFunction(My_Component_Mass_Rate_Function)

Additionally, a method is created which returns true or false depending on whether the method has been attached to a function yet, e.g.

myComponent%massIsAttached ()
myComponent%massSetIsAttached ()
myComponent%massRateIsAttached()
bindsTo Specifies to which level in the class hierarchy set, get and rate methods should be bound. Normally, these are bound to the component implementation itself. However, it can be useful to specify a binding of “top” to bind to the base nodeComponent class to make these methods interoperable with properties of other classes (see the discussion of the makeGeneric element above).

output [Optional] If present, the property will be included in the GALACTICUS output file. The following attributes control the details of that output:

- unitsInSI The units of the output quantity in the SI system.
- comment A comment to be included with the HDF5 dataset for this property.
- condition A statement which must evaluate to true or false and which will be used to determine if the property will be output. The present output time for is available as time. In the case of an array property the construct “{i}” can be used to pass the index of the element for which the condition should be evaluated.
- modules A comma-separated list of any modules required to perform the output (e.g. modules which contain functions or values that are used).

Additional attributes are required for array properties:

- labels This can be an array, declared as “[L_1,...,L_N]”, specifying the suffix to be added to the property name for each component of the array in the output, or a function which returns the suffix. In the case of a function the construct “{i}” can be used to pass the index of the element for which the suffix is required.
- count A statement which evaluates the number of elements to be output (i.e. the length of the array).

isVirtual [Optional] If present and set to “yes”, this property is a virtual property. A virtual property has no data associated with it and must supply its own functions for getting, setting and adjusting its rate of change (if allowed by the property’s attributes). Virtual properties are used for quantities which are derived from actual properties of the component implementation (for example, a star formation rate could be a virtual property if it is derived from an actual gas mass property) or for adjusting the rates of several actual properties simultaneously.

getFunction [Optional] Specifies the function to be used for getting the value of the property, overriding the default get function. The function must be included in the Galacticus_Nodes module by use of the functions element described below. Note that this function, by virtue of its privileged access to the internal structure of node components, can access the value of the data associated with the property using:

```
myComponent%<property>Data%value
```

setFunction [Optional] The same as getFunction but defines a function to set the value of the property.

classDefault [Optional] Specifies the default value for this property if the component class has not been created (i.e. has no specific implementation yet). The content of this element gives the default value (which can be a scalar, an array, a function, etc.). Additional, optional attributes control the use of this element:

- modules Specifies a comma-separated list of modules which are required to set the default values (e.g. modules which contain the value or function to be used).
- count For array properties whose size is not known at compile-time, it is possible to specify a function which will return the appropriate size of the array at run-time. The scalar default value given in the classDefault element will then be replicated the appropriate number of times.
17.3. Components

bindings [Optional] Contains an array of binding elements which specify functions to bind to this implementation. Each member binding has the following structure:

- **method** The name of the bound method, such that the function can be accessed using
  ```
  myComponent%<method>(...) 
  ```

- **function** The function to which the method should be bound. (This function must be included in the Galacticus_Nodes module by use of the functions element described below.

- **type** The type of function.

  - **bindsTo** Specifies where this method should be bound. “component” specifies binding to the specific implementation of this component class, “componentClass” specifies binding to the component class, while “top” specifies binding to the base nodeComponent class.

functions [Optional] Contains the name of a file which will be included into the Galacticus_Nodes module. This file can contain functions which will be bound to this implementation. By virtue of being included in the Galacticus_Nodes module these functions have privileged access to the internal structure of all node component objects.

Component Initialization

Initialization of a component module (if necessary, for example, to read parameters or allocate workspace) can occur at a number of different points in the execution of GALACTICUS. Providing initialization occurs in advance of any calculations then any point is acceptable. One possibility is simply to call an initialization function at the head of all functions defined in the component module. This initialization function should return immediately if it has already been called (to avoid duplicate initialization). Another option is to use the mergerTreePreTreeConstructionTask event (see §17.4.3) to perform initialization just before merger trees are constructed (the initialization function must again return immediately if it has been previously called).

 Optionally, a component may include a mergerTreeEvolveThreadInitialize directive, which gives the name of a subroutine in its unitName element. The routine specified by mergerTreeEvolveThreadInitialize is called by all threads prior to merger tree evolution, and can therefore be used to perform any “per thread” initialization. Note that this routine will be called many times during a given GALACTICUS run—it is the responsibility of the routine to ensure that it performs any initialization only once.

Component Access, Creation and Destruction

When a node is created, it initially contains no components. A component must therefore create itself on the fly as needed. Typically, a component is first created when an attempt is made to set a property value, or to adjust the rate of change of a property value or in response to some event (e.g. a satellite component may be created in response to a node merging with a larger node). Requests for property values frequently do not require that the component exist, as a zero value can often be returned instead\(^2\).

To access a component from a node, use:

```
myComponent => thisNode%<class>({instance=<N>,autoCreate=create})
```

where **class** is the component class required, the optional **instance** argument requests a specific instance of the component (relevant if the node contains more than one of a particular component, e.g. if it contains two supermassive black holes for example; if no instance is specified the first instance will be returned), and the **autoCreate** option specifies whether or not the component should be automatically created (assuming it does not already exist). **autoCreate=true** should be used to create components initially.

A component of a node can be destroyed using:

\(^2\) Or some other value if a classDefault has been specified (see §17.3.3).
call thisNode%<class>Destroy()

**Component Methods**

Component implementations optionally provide functions to get and set their properties (and to set the rate of change of evolvable properties) so that other components and functions within GALACTICUS can interact with them in a way that is independent of the specific component implementation chosen. To permit this, GALACTICUS creates functions for each property to access it in all permitted ways\(^3\). For example, the exponential implementation of the disk component class has a "massStellar" property defined by:

```
<method>
  <name>massStellar</name>
  <type>real</type>
  <rank>0</rank>
  <attributes isSettable="true" isGettable="true" isEvolvable="true" />
</method>
```

This causes GALACTICUS to define several functions bound to the nodeComponentDisk class:

- **massStellarIsSettable** Returns true if this property is settable;
- **massStellarIsGettable** Returns true if this property is gettable;
- **massStellarSet** Sets the value of this property to the supplied argument;
- **massStellarGet** Gets the value of this property;
- **massStellarRate** Cumulates its argument to the rate of change of this property;
- **massStellarScale** Sets the absolute scale for this property used in ODE error control;

along with several others used internally for output, serialization etc.

**Component Evolution**

All component properties which have an isEvolvable attribute set to true are included in GALACTICUS’s ODE solver as the node is evolved forward in time. As described in §17.3.3, GALACTICUS will create two functions that permit the rate of change of a property adjusted and for the absolute scale used in ODE error control to be set.

A “rate compute” function should be defined to perform any calculations necessary to determine the rate of change of the property and adjust the rate appropriately. Below is an example of the rate compute subroutine for the stellar mass property of the exponential disk component, with only the basic structure shown:

```
!# <rateComputeTask>
!# <unitName>Node_Component_Disk_Exponential_Rate_Compute</unitName>
!# </rateComputeTask>
subroutine Node_Component_Disk_Exponential_Rate_Compute(thisNode , &
   &interrupt , interruptProcedure)
   implicit none
   type (treeNode ) , pointer , intent(inout) :: thisNode
```

\(^3\)Additionally, C wrappers are generated to the get methods for real scalar properties. See §16.4 for a discussion of includeing C code within GALACTICUS.
17.3. Components

logical procedure( ), pointer, intent(inout) :: interrupt
&interruptProcedure
class (nodeComponentDisk ), pointer :: &thisDiskComponent

! Get the disk and check that it is of our class.
thisDiskComponent => thisNode%disk()
select type (thisDiskComponent)
class is (nodeComponentDiskExponential)

    call thisDiskComponent%massStellarRate(stellarMassRate)

end select
return
end subroutine Node_Component_Disk_Exponential_Rate_Comput

Here, we get the disk component and check that it is of the exponential variety. If it is, we compute the rates of change for one or more properties and then adjust their rates appropriately. If multiple instances of a component are used then the rate compute function should loop over all instances and adjust rates appropriately.

When evolving ODEs the ODE solver aims to keep the error on property $i$ below

$$D_i = \epsilon_{abs}s_i + \epsilon_{rel}|y_i|,$$

where $\epsilon_{abs} = \text{[odeToleranceAbsolute]}$, $\epsilon_{rel} = \text{[odeToleranceRelative]}$, $y_i$ is the value of property $i$ and $s_i$ is a scaling factor which controls the absolute tolerance for this property. By default, $s_i = 1$, but this can be changed for a component utilizing the $\text{scaleSetTask}$ directive. This allows a function to be called in which the component sets suitable scale factors for each of its properties prior to any ODE evolution being carried out. This can be very useful, for example, in cases where two components are coupled. Consider a case where a disk is transferring material to a spheroid via a bar instability. If the disk is orders of magnitude more massive that the spheroid then the rate of mass transfer can be very high (i.e. $\dot{y}/y$ for the spheroid will be large). With just a relative tolerance (i.e. the $\epsilon_{rel}|y_i|$ term) this would require very short timesteps for the spheroid. However, in such cases we don’t care about such tiny tolerances for the spheroid (since it will grow to be substantially more massive). Therefore, it may be appropriate to set $s_i$ to be equal to the sum of the disk and spheroid properties for example. The scale set directive and associated subroutine should follow this template:

```fortran
!# <scaleSetTask>
!# <unitName>Node_Component_Disk_Exponential_Scale_Set</unitName>
!# </scaleSetTask>
subroutine Node_Component_Disk_Exponential_Scale_Set(thisNode)
    implicit none
    type (treeNode ), pointer, intent(inout) :: thisNode
class(nodeComponentDisk), pointer :: thisDiskComponent

    ! Get the disk component.
    thisDiskComponent => thisNode%disk()
    ! Check if an exponential disk component exists.
    select type (thisDiskComponent)
class is (nodeComponentDiskExponential)
```

719
17. Adding New Methods

... call thisDiskComponent%massStellarScale(massScale) ...
end select
return
end subroutine Node_Component_Disk_Exponential_Scale_Set

Sensible choices for the $s_i$ factors can significantly speed-up execution of GALACTICUS.

Evolution Interrupts

It is often necessary to interrupt the smooth ODE evolution of a node in GALACTICUS. This can happen if, for example, a galaxy mergers with another galaxy (in which case the merger must be processed prior to further evolution) or if a component must be created before evolution can continue. The rate adjust and rate compute subroutines allow for interrupts to be flagged via their interrupt and interruptProcedure arguments. If an interrupt is required then interrupt should be set to true, while interruptProcedure should be set to point to a procedure which will handle the interrupt. Then, providing no other interrupt occurred earlier, the evolution will be stopped and the interrupt procedure called before evolution is continued.

An interrupt procedure should have the form:

```fortran
subroutine My_Interrupt_Procedure(thisNode)
  implicit none
  type(treeNode), pointer, intent(inout) :: thisNode

  ! Do whatever needs to be done to handle the interrupt.

  return
end subroutine My_Interrupt_Procedure
```

17.4. Existing Method Types

17.4.1. Functions

Functions implement basic calculations (e.g. computing the power spectrum).

Accretion Disk Spectra

Additional implementations for accretion disk spectra are added using the accretionDiskSpectra class. The implementation should be placed in a file containing the directive:

```fortran
!# <accretionDiskSpectra name="accretionDiskSpectraMyImplementation">
!# <description>A short description of the implementation.</description>
!# </accretionDiskSpectra>
```

where MyImplementation is an appropriate name for the implementation. This file should be treated as a regular Fortran module, but without the initial module and final end module lines. That is, it may contain use statements and variable declarations prior to the contains line, and should contain all functions required by the implementation after that line. Function names should begin with accretionDiskSpectraMyImplementation. The file must define a type that extends the accretionDiskSpectraClass class (or extends another type which is itself an extension of the accretionDiskSpectraClass class),
17.4. Existing Method Types

containing any data needed by the implementation along with type-bound functions required by the implementation. The following type-bound functions are required (unless inherited from the parent type):

**spectrum** Returns the spectrum (in units of $L_{\odot}$ Hz$^{-1}$) of the accretion disk at the given wavelength (in units of Å) for node. Must have the following interface:

```fortran
double precision function myImplementationSpectrum(self,node,wavelength)
  class (accretionDiskSpectraClass), intent(inout) :: self
  type (treeNode), intent(inout) :: node
  double precision, intent(in) :: wavelength
end double precision function myImplementationSpectrum
```

**isFinalizable** Return true if this object can be finalized. A default implementation exists. If overridden the following interface must be used:

```fortran
logical function myImplementationIsFinalizable(self)
  class(accretionDiskSpectraClass), intent(inout) :: self
end logical function myImplementationIsFinalizable
```

Existing implementations are:

- **accretionDiskSpectraFile** Accretion disk spectra are interpolated from tables read from file.
- **accretionDiskSpectraHopkins2007** Accretion disk spectra using the model of Hopkins et al. [2007].

**Accretion Onto Halos**

Additional implementations for accretion onto halos are added using the **accretionHalo** class. The implementation should be placed in a file containing the directive:

```
!# <accretionHalo name="accretionHaloMyImplementation">
!# <description>A short description of the implementation.</description>
!# </accretionHalo>
```

where **MyImplementation** is an appropriate name for the implementation. This file should be treated as a regular Fortran module, but without the initial **module** and final **end module** lines. That is, it may contain use statements and variable declarations prior to the contains line, and should contain all functions required by the implementation after that line. Function names should begin with **accretion-HaloMyImplementation**. The file must define a type that extends the **accretionHaloClass** class (or extends another type which is itself an extension of the **accretionHaloClass** class), containing any data needed by the implementation along with type-bound functions required by the implementation. The following type-bound functions are required (unless inherited from the parent type):

**accretedMass** Returns the mass (in units of $M_{\odot}$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes. Must have the following interface:

```fortran
double precision function myImplementationAccretedMass(self,node,accretionMode)
  class (accretionHaloClass), intent(inout) :: self
  type (treeNode), intent(inout), pointer :: node
  integer, intent(in) :: accretionMode
end double precision function myImplementationAccretedMass
```

**accretedMassChemicals** Returns the mass of chemicals (in units of $M_{\odot}$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes. Must have the following interface:
17. Adding New Methods

type(chemicalAbundances) function myImplementationAccretedMassChemicals(self, node, & accretionMode)
    class (accretionHaloClass), intent(inout) :: self
    type (treeNode  ), intent(inout), pointer :: node
    integer , intent(in ) :: accretionMode
end type(chemicalAbundances) function myImplementationAccretedMassChemicals

accretedMassMetals Returns the mass of metals (in units of $M\odot$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes. Must have the following interface:

type(abundances) function myImplementationAccretedMassMetals(self, node, accretionMode)
    class (accretionHaloClass), intent(inout) :: self
    type (treeNode  ), intent(inout), pointer :: node
    integer , intent(in) :: accretionMode
end type(abundances) function myImplementationAccretedMassMetals

accretionRate Returns the rate (in units of $M\odot$ Gyr$^{-1}$) of accretion of mass from the IGM onto node in the given accretionMode. Must have the following interface:

double precision function myImplementationAccretionRate(self, node, accretionMode)
    class (accretionHaloClass), intent(inout) :: self
    type (treeNode  ), intent(inout), pointer :: node
    integer , intent(in) :: accretionMode
end double precision function myImplementationAccretionRate

accretionRateChemicals Returns the rate (in units of $M\odot$ Gyr$^{-1}$) of accretion of chemicals from the IGM onto node in the given accretionMode. Must have the following interface:

type(chemicalAbundances) function myImplementationAccretionRateChemicals(self, node, & accretionMode)
    class (accretionHaloClass), intent(inout) :: self
    type (treeNode  ), intent(inout), pointer :: node
    integer , intent(in) :: accretionMode
end type(chemicalAbundances) function myImplementationAccretionRateChemicals

accretionRateMetals Returns the rate (in units of $M\odot$ Gyr$^{-1}$) of accretion of metals from the IGM onto node in the given accretionMode. Must have the following interface:

type(abundances) function myImplementationAccretionRateMetals(self, node, accretionMode)
    class (accretionHaloClass), intent(inout) :: self
    type (treeNode  ), intent(inout), pointer :: node
    integer , intent(in) :: accretionMode
end type(abundances) function myImplementationAccretionRateMetals

failedAccretedMass Returns the mass (in units of $M\odot$) of that failed to accrete from the IGM onto node in the given accretionMode. Used to initialize nodes. Must have the following interface:

double precision function myImplementationFailedAccretedMass(self, node, accretionMode)
    class (accretionHaloClass), intent(inout) :: self
    type (treeNode  ), intent(inout), pointer :: node
    integer , intent(in) :: accretionMode
end double precision function myImplementationFailedAccretedMass

failedAccretionRate Returns the rate (in units of $M\odot$ Gyr$^{-1}$) of failed accretion of mass from the IGM onto node in the given accretionMode. Must have the following interface:
double precision function myImplementationFailedAccretionRate(self, node, accretionMode)
class (accretionHaloClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: node
integer , intent(in ) :: accretionMode
end double precision function myImplementationFailedAccretionRate

isFinalizable Return true if this object can be finalized. A default implementation exists. If overridden
the following interface must be used:

logical function myImplementationIsFinalizable(self)
class(accretionHaloClass), intent(inout) :: self
end logical function myImplementationIsFinalizable

Existing implementations are:

accretionHaloNull Accretion onto halos assuming no accretion.

accretionHaloSimple Accretion onto halos using simple truncation to mimic the effects of reionization.

accretionHaloColdMode Accretion onto halos using simple truncation to mimic the effects of reionization
and accounting for cold mode accretion.

accretionHaloNaozBarkana2007 Accretion onto halos using filtering mass of the IGM calculated from
an equation from Naoz and Barkana [2007].

Cosmology Functions

Additional implementations for cosmology functions are added using the cosmologyFunctions class. The
implementation should be placed in a file containing the directive:

!# <cosmologyFunctions name="cosmologyFunctionsMyImplementation">
!# <description>A short description of the implementation.</description>
!# </cosmologyFunctions>

where MyImplementation is an appropriate name for the implementation. This file should be treated as
a regular Fortran module, but without the initial module and final end module lines. That is, it may
contain use statements and variable declarations prior to the contains line, and should contain all
functions required by the implementation after that line. Function names should begin with cosmology-
FunctionsMyImplementation. The file must define a type that extends the cosmologyFunctionsClass
class (or extends another type which is itself an extension of the cosmologyFunctionsClass class),
containing any data needed by the implementation along with type-bound functions required by the
implementation. The following type-bound functions are required (unless inherited from the parent
type):

comovingVolumeElementRedshift Returns the differential comoving volume element \(dV/dz = r_c^2(t)H^{-1}(t)\)
(where \(r_c\) is the comoving distance to time \(t\) and \(H(t)\) is the Hubble parameter at that time) for
unit solid angle at the specified time. A default implementation exists. If overridden the following
interface must be used:

double precision function myImplementationComovingVolumeElementRedshift(self, time)
class (cosmologyFunctionsClass), intent(inout) :: self
double precision , intent(in ) :: time
end double precision function myImplementationComovingVolumeElementRedshift
17. Adding New Methods

comovingVolumeElementTime Returns the differential comoving volume element $dV/dt = r_c^2(t)a(t)$ (where $r_c$ is the comoving distance to time $t$ and $a(t)$ is the expansion at that time) for unit solid angle at the specified time. A default implementation exists. If overridden the following interface must be used:

```fortran
double precision function myImplementationComovingVolumeElementTime(self,time)
    class (cosmologyFunctionsClass), intent(inout) :: self
    double precision , intent(in ) :: time
end double precision function myImplementationComovingVolumeElementTime
```

cosmicTime Return the cosmological age at the given expansion factor. Must have the following interface:

```fortran
double precision function myImplementationCosmicTime(self,expansionFactor,&
    collapsingPhase)
    class (cosmologyFunctionsClass), intent(inout) :: self
    double precision , intent(in ) :: expansionFactor
    logical , intent(in ), optional :: collapsingPhase
end double precision function myImplementationCosmicTime
```

cosmicTimeIsValid Returns true if the given cosmic time is valid one for this cosmology. Must have the following interface:

```fortran
logical function myImplementationCosmicTimeIsValid(self,time)
    class (cosmologyFunctionsClass), intent(inout) :: self
    double precision , intent(in ) :: time
end logical function myImplementationCosmicTimeIsValid
```

densityScalingEarlyTime Compute the scaling of density with expansion factor at early times in the universe. Must have the following interface:

```fortran
subroutine myImplementationDensityScalingEarlyTime(self,dominateFactor,densityPower , &
    expansionFactorDominant,OmegaDominant)
    class (cosmologyFunctionsClass), intent(inout) :: self
    double precision , intent(in ) :: dominateFactor
    double precision , intent(out) :: densityPower, &
    double precision , intent(out) :: OmegaDominant
    double precision , intent(out), optional :: OmegaDominant
end subroutine myImplementationDensityScalingEarlyTime
```

distanceAngular Return the angular diameter distance to the given cosmic time. Must have the following interface:

```fortran
double precision function myImplementationDistanceAngular(self,time)
    class (cosmologyFunctionsClass), intent(inout) :: self
    double precision , intent(in ) :: time
end double precision function myImplementationDistanceAngular
```

distanceComoving Return the comoving distance to the given cosmic time. Must have the following interface:

```fortran
double precision function myImplementationDistanceComoving(self,time)
    class (cosmologyFunctionsClass), intent(inout) :: self
    double precision , intent(in ) :: time
end double precision function myImplementationDistanceComoving
17.4. Existing Method Types

distanceComovingConvert Convert between different measures of comoving distance. Must have the following interface:

\[
\text{double precision function myImplementationDistanceComovingConvert(self, output, \\
& \text{distanceModulus}, \text{redshift})}
\]

\text{class (cosmologyFunctionsClass), intent(inout) :: self}

\text{integer, intent(in) :: output}

\text{double precision, intent(in), optional :: distanceModulus, \\
& \text{redshift}}

\text{end double precision function myImplementationDistanceComovingConvert}

distanceLuminosity Return the luminosity distance to the given cosmic time. Must have the following interface:

\[
\text{double precision function myImplementationDistanceLuminosity(self, time)}
\]

\text{class (cosmologyFunctionsClass), intent(inout) :: self}

\text{double precision, intent(in) :: time}

\text{end double precision function myImplementationDistanceLuminosity}

dominationEpochMatter Compute the epoch at which matter dominates over other forms of energy by a given factor. Must have the following interface:

\[
\text{double precision function myImplementationDominationEpochMatter(self, dominateFactor)}
\]

\text{class (cosmologyFunctionsClass), intent(inout) :: self}

\text{double precision, intent(in) :: dominateFactor}

\text{end double precision function myImplementationDominationEpochMatter}

ePOCHTIME Convenience function that returns the time corresponding to an epoch specified by time or expansion factor. A default implementation exists. If overridden the following interface must be used:

\[
\text{double precision function myImplementationEpochTime(self, time, \\
& \text{expansionFactor, collapsingPhase})}
\]

\text{class (cosmologyFunctionsClass), intent(inout) :: self}

\text{double precision, intent(in), optional :: time, \\
& \text{expansionFactor}}

\text{logical, intent(in), optional :: collapsingPhase}

\text{end double precision function myImplementationEpochTime}

equalityEpochMatterCurvature Return the epoch of matter-curvature magnitude equality (either expansion factor or cosmic time). Must have the following interface:

\[
\text{double precision function myImplementationEqualityEpochMatterCurvature(self, requestType, \\
& \text{requestType})}
\]

\text{class (cosmologyFunctionsClass), intent(inout) :: self}

\text{integer, intent(in), optional :: requestType}

\text{end double precision function myImplementationEqualityEpochMatterCurvature}

equalityEpochMatterDarkEnergy Return the epoch of matter-dark energy magnitude equality (either expansion factor or cosmic time). Must have the following interface:

\[
\text{double precision function myImplementationEqualityEpochMatterDarkEnergy(self, requestType, \\
& \text{requestType})}
\]

\text{class (cosmologyFunctionsClass), intent(inout) :: self}

\text{integer, intent(in), optional :: requestType}

\text{end double precision function myImplementationEqualityEpochMatterDarkEnergy}
17. Adding New Methods

equationOfStateDarkEnergy  HASH(0x1989a30)  Must have the following interface:

\[
\text{double precision function } \text{myImplementationEquationOfStateDarkEnergy}(\text{self}, \text{time}, \& \text{expansionFactor})
\]

\[
\begin{align*}
\text{class} & \quad \text{(cosmologyFunctionsClass), intent(inout)} : : \text{self} \\
\text{double precision} & \quad \text{(in )}, \text{optional : : time}, \& \\
& \quad \text{& expansionFactor}
\end{align*}
\]

\[
\text{end double precision function } \text{myImplementationEquationOfStateDarkEnergy}
\]

expansionFactor  Returns the expansion factor at cosmological time \( \text{time} \). Must have the following interface:

\[
\text{double precision function } \text{myImplementationExpansionFactor}(\text{self}, \text{time})
\]

\[
\begin{align*}
\text{class} & \quad \text{(cosmologyFunctionsClass), intent(inout)} : : \text{self} \\
\text{double precision} & \quad \text{(in )} : : \text{time}
\end{align*}
\]

\[
\text{end double precision function } \text{myImplementationExpansionFactor}
\]

expansionFactorFromRedshift  Returns expansion factor given a redshift. A default implementation exists. If overridden the following interface must be used:

\[
\text{double precision function } \text{myImplementationExpansionFactorFromRedshift}(\text{self}, \text{redshift})
\]

\[
\begin{align*}
\text{class} & \quad \text{(cosmologyFunctionsClass), intent(inout)} : : \text{self} \\
\text{double precision} & \quad \text{(in )} : : \text{redshift}
\end{align*}
\]

\[
\text{end double precision function } \text{myImplementationExpansionFactorFromRedshift}
\]

expansionFactorIsValid  Returns true if the given expansion factor is valid one for this cosmology. Must have the following interface:

\[
\text{logical function } \text{myImplementationExpansionFactorIsValid}(\text{self}, \text{expansionFactor})
\]

\[
\begin{align*}
\text{class} & \quad \text{(cosmologyFunctionsClass), intent(inout)} : : \text{self} \\
\text{double precision} & \quad \text{(in )} : : \text{expansionFactor}
\end{align*}
\]

\[
\text{end logical function } \text{myImplementationExpansionFactorIsValid}
\]

expansionRate  Returns the cosmological expansion rate, \( \dot{a}/a \) at expansion factor \( \text{expansionFactor} \). Must have the following interface:

\[
\text{double precision function } \text{myImplementationExpansionRate}(\text{self}, \text{expansionFactor})
\]

\[
\begin{align*}
\text{class} & \quad \text{(cosmologyFunctionsClass), intent(inout)} : : \text{self} \\
\text{double precision} & \quad \text{(in )} : : \text{expansionFactor}
\end{align*}
\]

\[
\text{end double precision function } \text{myImplementationExpansionRate}
\]

exponentDarkEnergy  HASH(0x198caa8)  Must have the following interface:

\[
\text{double precision function } \text{myImplementationExponentDarkEnergy}(\text{self}, \text{time}, \text{expansionFactor})
\]

\[
\begin{align*}
\text{class} & \quad \text{(cosmologyFunctionsClass), intent(inout)} : : \text{self} \\
\text{double precision} & \quad \text{(in )}, \text{optional : : time}, \& \\
& \quad \text{& expansionFactor}
\end{align*}
\]

\[
\text{end double precision function } \text{myImplementationExponentDarkEnergy}
\]

hubbleParameterEpochal  Returns the Hubble parameter at the requested cosmological time, \( \text{time} \), or expansion factor, \( \text{expansionFactor} \). Must have the following interface:
17.4. Existing Method Types

double precision function myImplementationHubbleParameterEpochal(self,time,&
& expansionFactor,collapsingPhase)
  class (cosmologyFunctionsClass), intent(inout) :: self
double precision , intent(in ), optional :: time, &
& expansionFactor
logical , intent(in ), optional :: collapsingPhase
end double precision function myImplementationHubbleParameterEpochal

hubbleParameterRateOfChange Returns the rate of change of the Hubble parameter at the requested cosmological time, \( \text{time} \), or expansion factor, \( \text{expansionFactor} \). Must have the following interface:

double precision function myImplementationHubbleParameterRateOfChange(self,time,&
& expansionFactor,collapsingPhase)
  class (cosmologyFunctionsClass), intent(inout) :: self
double precision , intent(in ), optional :: time, &
& expansionFactor
logical , intent(in ), optional :: collapsingPhase
end double precision function myImplementationHubbleParameterRateOfChange

omegaDarkEnergyEpochal Return the dark energy density parameter at expansion factor \( \text{expansionFactor} \). Must have the following interface:

double precision function myImplementationOmegaDarkEnergyEpochal(self,time , &
& expansionFactor,collapsingPhase)
  class (cosmologyFunctionsClass), intent(inout) :: self
double precision , intent(in ), optional :: time, &
& expansionFactor
logical , intent(in ), optional :: collapsingPhase
end double precision function myImplementationOmegaDarkEnergyEpochal

omegaMatterEpochal Return the matter density parameter at expansion factor \( \text{expansionFactor} \). Must have the following interface:

double precision function myImplementationOmegaMatterEpochal(self,time , &
& expansionFactor,collapsingPhase)
  class (cosmologyFunctionsClass), intent(inout) :: self
double precision , intent(in ), optional :: time, &
& expansionFactor
logical , intent(in ), optional :: collapsingPhase
end double precision function myImplementationOmegaMatterEpochal

omegaMatterRateOfChange Return the rate of change of the matter density parameter at expansion factor \( \text{expansionFactor} \). Must have the following interface:

double precision function myImplementationOmegaMatterRateOfChange(self,time , &
& expansionFactor,collapsingPhase)
  class (cosmologyFunctionsClass), intent(inout) :: self
double precision , intent(in ), optional :: time, &
& expansionFactor
logical , intent(in ), optional :: collapsingPhase
end double precision function myImplementationOmegaMatterRateOfChange

redshiftFromExpansionFactor Returns redshift for a given expansion factor. A default implementation exists. If overridden the following interface must be used:
17. Adding New Methods

double precision function myImplementationRedshiftFromExpansionFactor(self, & expansionFactor)
  class (cosmologyFunctionsClass), intent(inout) :: self
double precision , intent(in ) :: expansionFactor
end double precision function myImplementationRedshiftFromExpansionFactor

**temperatureCMBEpochal** Return the temperature of the cosmic microwave background at **expansionFactor**. Must have the following interface:

double precision function myImplementationTemperatureCMBEpochal(self, time , & expansionFactor, collapsingPhase)
  class (cosmologyFunctionsClass), intent(inout) :: self
double precision , intent(in ), optional :: time, & expansionFactor
logical , intent(in ), optional :: collapsingPhase
end double precision function myImplementationTemperatureCMBEpochal

**timeAtDistanceComoving** Return the cosmic time corresponding to the given comovingDistance. Must have the following interface:

double precision function myImplementationTimeAtDistanceComoving(self, comovingDistance)
  class (cosmologyFunctionsClass), intent(inout) :: self
double precision , intent(in ) :: comovingDistance
end double precision function myImplementationTimeAtDistanceComoving

**stateStore** Store the state of the object to file. A default implementation exists. If overridden the following interface must be used:

subroutine myImplementationStateStore(self, stateFile, fgslStateFile)
  class (cosmologyFunctionsClass), intent(inout) :: self
integer , intent(in ) :: stateFile
type (fgsl_file ), intent(in ) :: fgslStateFile
end subroutine myImplementationStateStore

**stateRestore** Restore the state of the object to file. A default implementation exists. If overridden the following interface must be used:

subroutine myImplementationStateRestore(self, stateFile, fgslStateFile)
  class (cosmologyFunctionsClass), intent(inout) :: self
integer , intent(in ) :: stateFile
type (fgsl_file ), intent(in ) :: fgslStateFile
end subroutine myImplementationStateRestore

**isFinalizable** Return true if this object can be finalized. A default implementation exists. If overridden the following interface must be used:

logical function myImplementationIsFinalizable(self)
  class (cosmologyFunctionsClass), intent(inout) :: self
end logical function myImplementationIsFinalizable

Existing implementations are:

cosmologyFunctionsMatterLambda Cosmological relations are computed assuming a universe that contains only matter and a cosmological constant. See §13.2.1.

cosmologyFunctionsMatterDarkEnergy Cosmological relations are computed assuming a universe that contains only matter and dark energy with an equation of state \( w(a) = w_0 + w_1 (1 - a) \). See §13.2.2.
Cosmological Parameters

Additional implementations for cosmological parameters are added using the `cosmologyParameters` class. The implementation should be placed in a file containing the directive:

```fortran
!# <cosmologyParameters name="cosmologyParametersMyImplementation">  
!# <description>A short description of the implementation.</description>  
!# </cosmologyParameters>
```

where `MyImplementation` is an appropriate name for the implementation. This file should be treated as a regular Fortran module, but without the initial `module` and final `end module` lines. That is, it may contain `use` statements and variable declarations prior to the `contains` line, and should contain all functions required by the implementation after that line. Function names should begin with `cosmologyParameters-MyImplementation`. The file must define a type that extends the `cosmologyParametersClass` class (or extends another type which is itself an extension of the `cosmologyParametersClass` class), containing any data needed by the implementation along with type-bound functions required by the implementation. The following type-bound functions are required (unless inherited from the parent type):

**HubbleConstant** Return the Hubble constant at the present day. The optional `units` argument specifies if the return value should be in units of \( \text{km/s/Mpc} \) (`unitsStandard`), \( \text{Gyr}^{-1} \) (`unitsTime`), or 100 \( \text{km/s/Mpc} \) (`unitsLittleH`). Must have the following interface:

```fortran
double precision function myImplementationHubbleConstant(self, units)
    class (cosmologyParametersClass), intent(inout) :: self
    integer , intent(in), optional :: units
end double precision function myImplementationHubbleConstant
```

**OmegaBaryon** Return the cosmological baryon density in units of the critical density at the present day. Must have the following interface:

```fortran
double precision function myImplementationOmegaBaryon(self)
    class (cosmologyParametersClass), intent(inout) :: self
end double precision function myImplementationOmegaBaryon
```

**OmegaCurvature** Return the cosmological curvature density in units of the critical density at the present day. Must have the following interface:

```fortran
double precision function myImplementationOmegaCurvature(self)
    class (cosmologyParametersClass), intent(inout) :: self
end double precision function myImplementationOmegaCurvature
```

**OmegaDarkEnergy** Return the cosmological dark energy density in units of the critical density at the present day. Must have the following interface:

```fortran
double precision function myImplementationOmegaDarkEnergy(self)
    class (cosmologyParametersClass), intent(inout) :: self
end double precision function myImplementationOmegaDarkEnergy
```

**OmegaMatter** Return the cosmological matter density in units of the critical density at the present day. Must have the following interface:

```fortran
double precision function myImplementationOmegaMatter(self)
    class (cosmologyParametersClass), intent(inout) :: self
end double precision function myImplementationOmegaMatter
```
17. Adding New Methods

OmegaRadiation  Return the cosmological radiation density in units of the critical density at the present day. Must have the following interface:

    double precision function myImplementationOmegaRadiation(self)
    class(cosmologyParametersClass), intent(inout) :: self
end double precision function myImplementationOmegaRadiation

densityCritical  Return the critical density at the present day in units of $M_\odot$/Mpc$^3$. Must have the following interface:

    double precision function myImplementationDensityCritical(self)
    class(cosmologyParametersClass), intent(inout) :: self
end double precision function myImplementationDensityCritical

temperatureCMB  Return the temperature of the cosmic microwave background radiation (in units of Kelvin) at the present day. Must have the following interface:

    double precision function myImplementationTemperatureCMB(self)
    class(cosmologyParametersClass), intent(inout) :: self
end double precision function myImplementationTemperatureCMB

isFinalizable  Return true if this object can be finalized. A default implementation exists. If overridden the following interface must be used:

    logical function myImplementationIsFinalizable(self)
    class(cosmologyParametersClass), intent(inout) :: self
end logical function myImplementationIsFinalizable

Existing implementations are:

cosmologyParametersSimple  Provides basic cosmological parameters: ($H_0, \Omega_M, \Omega_\Lambda, \Omega_b, T_{CMB}$). Also provides derived quantities ($\Omega_K, \Omega_r, \rho_{crit}$).

Dark Matter Halo Scales

Additional implementations for dark matter halo scales are added using the darkMatterHaloScale class. The implementation should be placed in a file containing the directive:

    !<darkMatterHaloScale name="darkMatterHaloScaleMyImplementation">
    !<description>A short description of the implementation.</description>
    !</darkMatterHaloScale>

where MyImplementation is an appropriate name for the implementation. This file should be treated as a regular Fortran module, but without the initial module and final end module lines. That is, it may contain use statements and variable declarations prior to the contains line, and should contain all functions required by the implementation after that line. Function names should begin with darkMatterHaloScaleMyImplementation. The file must define a type that extends the darkMatterHaloScaleClass class (or extends another type which is itself an extension of the darkMatterHaloScaleClass class), containing any data needed by the implementation along with type-bound functions required by the implementation. The following type-bound functions are required (unless inherited from the parent type):

dynamicalTimescale  The characteristic dynamical timescale of a dark matter halo. Must have the following interface:
double precision function myImplementationDynamicalTimescale(self,thisNode)
class(darkMatterHaloScaleClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: thisNode
end double precision function myImplementationDynamicalTimescale

meanDensity The mean density of a dark matter halo. Must have the following interface:

double precision function myImplementationMeanDensity(self,thisNode)
class(darkMatterHaloScaleClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: thisNode
end double precision function myImplementationMeanDensity

meanDensityGrowthRate The growth rate of the mean density of a dark matter halo. Must have the following interface:

double precision function myImplementationMeanDensityGrowthRate(self,thisNode)
class(darkMatterHaloScaleClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: thisNode
end double precision function myImplementationMeanDensityGrowthRate

virialRadius The virial radius of a dark matter halo. Must have the following interface:

double precision function myImplementationVirialRadius(self,thisNode)
class(darkMatterHaloScaleClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: thisNode
end double precision function myImplementationVirialRadius

virialRadiusGrowthRate The growth rate of the virial radius of a dark matter halo. Must have the following interface:

double precision function myImplementationVirialRadiusGrowthRate(self,thisNode)
class(darkMatterHaloScaleClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: thisNode
end double precision function myImplementationVirialRadiusGrowthRate

virialTemperature The virial temperature of a dark matter halo. Must have the following interface:

double precision function myImplementationVirialTemperature(self,thisNode)
class(darkMatterHaloScaleClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: thisNode
end double precision function myImplementationVirialTemperature

virialVelocity The virial velocity of a dark matter halo. Must have the following interface:

double precision function myImplementationVirialVelocity(self,thisNode)
class(darkMatterHaloScaleClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: thisNode
end double precision function myImplementationVirialVelocity

virialVelocityGrowthRate The growth rate of the virial velocity of a dark matter halo. Must have the following interface:

double precision function myImplementationVirialVelocityGrowthRate(self,thisNode)
class(darkMatterHaloScaleClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: thisNode
end double precision function myImplementationVirialVelocityGrowthRate
stateStore Store the state of the object to file. A default implementation exists. If overridden the following interface must be used:

```fortran
subroutine myImplementationStateStore(self, stateFile, fgslStateFile)
  class (darkMatterHaloScaleClass), intent(inout) :: self
  integer , intent(in ) :: stateFile
  type (fgsl_file ), intent(in ) :: fgslStateFile
end subroutine myImplementationStateStore
```

stateRestore Restore the state of the object to file. A default implementation exists. If overridden the following interface must be used:

```fortran
subroutine myImplementationStateRestore(self, stateFile, fgslStateFile)
  class (darkMatterHaloScaleClass), intent(inout) :: self
  integer , intent(in ) :: stateFile
  type (fgsl_file ), intent(in ) :: fgslStateFile
end subroutine myImplementationStateRestore
```

calculationReset Reset the calculation state of the object. A default implementation exists. If overridden the following interface must be used:

```fortran
subroutine myImplementationCalculationReset(self, thisNode)
  class(darkMatterHaloScaleClass), intent(inout) :: self
  type (treeNode ), intent(inout), pointer :: thisNode
end subroutine myImplementationCalculationReset
```

isFinalizable Return true if this object can be finalized. A default implementation exists. If overridden the following interface must be used:

```fortran
logical function myImplementationIsFinalizable(self)
  class(darkMatterHaloScaleClass), intent(inout) :: self
end logical function myImplementationIsFinalizable
```

Existing implementations are:

darkMatterHaloScaleVirialDensityContrastDefinition Dark matter halo scales derived from virial density contrasts.

**Dark Matter Halo Profiles**

Additional implementations for dark matter halo profiles are added using the darkMatterProfile class. The implementation should be placed in a file containing the directive:

```fortran
!# <darkMatterProfile name="darkMatterProfileMyImplementation">
!# <description>A short description of the implementation.</description>
!# </darkMatterProfile>
```

where MyImplementation is an appropriate name for the implementation. This file should be treated as a regular Fortran module, but without the initial module and final end module lines. That is, it may contain use statements and variable declarations prior to the contains line, and should contain all functions required by the implementation after that line. Function names should begin with darkMatterProfileMyImplementation. The file must define a type that extends the darkMatterProfileClass class (or extends another type which is itself an extension of the darkMatterProfileClass class), containing any data needed by the implementation along with type-bound functions required by the implementation. The following type-bound functions are required (unless inherited from the parent type):
circularVelocity Returns the circular velocity (in km/s) in the dark matter profile of node at the given radius (given in units of Mpc). Must have the following interface:

```fortran
double precision function myImplementationCircularVelocity(self,node,radius)
    class (darkMatterProfileClass), intent(inout) :: self
    type (treeNode ), intent(inout), pointer :: node
    double precision , intent(in ) :: radius
end double precision function myImplementationCircularVelocity
```

circularVelocityMaximum Returns the maximum circular velocity (in km/s) in the dark matter profile of node. Must have the following interface:

```fortran
double precision function myImplementationCircularVelocityMaximum(self,node)
    class(darkMatterProfileClass), intent(inout) :: self
    type (treeNode ), intent(inout), pointer :: node
end double precision function myImplementationCircularVelocityMaximum
```

density Returns the density (in $M_\odot \, \text{Mpc}^{-3}$) in the dark matter profile of node at the given radius (given in units of Mpc). Must have the following interface:

```fortran
double precision function myImplementationDensity(self,node,radius)
    class (darkMatterProfileClass), intent(inout) :: self
    type (treeNode ), intent(inout), pointer :: node
    double precision , intent(in ) :: radius
end double precision function myImplementationDensity
```

enclosedMass Returns the enclosed mass (in $M_\odot$) in the dark matter profile of node at the given radius (given in units of Mpc). for the given node. Must have the following interface:

```fortran
double precision function myImplementationEnclosedMass(self,node,radius)
    class (darkMatterProfileClass), intent(inout) :: self
    type (treeNode ), intent(inout), pointer :: node
    double precision , intent(in ) :: radius
end double precision function myImplementationEnclosedMass
```

energy Return the total energy for the given node in units of $M_\odot \, \text{km}^2 \, \text{s}^{-1}$. Must have the following interface:

```fortran
double precision function myImplementationEnergy(self,node)
    class(darkMatterProfileClass), intent(inout) :: self
    type (treeNode ), intent(inout), pointer :: node
end double precision function myImplementationEnergy
```

energyGrowthRate Returns the rate of chance of the total energy of node in units of $M_\odot \, \text{km}^2 \, \text{s}^{-1} \, \text{Gyr}^{-1}$. Must have the following interface:

```fortran
double precision function myImplementationEnergyGrowthRate(self,node)
    class(darkMatterProfileClass), intent(inout) :: self
    type (treeNode ), intent(inout), pointer :: node
end double precision function myImplementationEnergyGrowthRate
```

freeFallRadiusIncreaseRate Returns the rate of increase of the freefall radius (in Mpc/Gyr) corresponding to the given time (in Gyr) in node. Must have the following interface:
17. Adding New Methods

**freefallRadius** Returns the freefall radius (in Mpc) corresponding to the given time (in Gyr) in node. Must have the following interface:

```fortran
double precision function myImplementationFreefallRadius(self,node,time)
  class (darkMatterProfileClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: node
double precision , intent(in ) :: time
end double precision function myImplementationFreefallRadius
```

**kSpace** Returns the normalized Fourier space density profile of the dark matter profile of node at the given waveNumber (given in units of Mpc$^{-1}$). Must have the following interface:

```fortran
double precision function myImplementationKSpace(self,node,wavenumber)
  class (darkMatterProfileClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: node
double precision , intent(in ) :: wavenumber
end double precision function myImplementationKSpace
```

**potential** Returns the gravitational potential (in $(km/s)^2$) in the dark matter profile of node at the given radius (given in units of Mpc). Must have the following interface:

```fortran
double precision function myImplementationPotential(self,node,radius,status)
  class (darkMatterProfileClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: node
double precision , intent(in ) :: radius
integer , intent( out), optional :: status
end double precision function myImplementationPotential
```

**radiusFromSpecificAngularMomentum** Returns the radius (in Mpc) in the dark matter profile of node at which the specific angular momentum of a circular orbit equals specificAngularMomentum (specified in units of km s$^{-1}$ Mpc. Must have the following interface:

```fortran
double precision function myImplementationRadiusFromSpecificAngularMomentum(self,node,& specificAngularMomentum)
  class (darkMatterProfileClass), intent(inout) :: self
type (treeNode ) , intent(inout), pointer :: node
double precision , intent(in ) :: & specificAngularMomentum
end double precision function myImplementationRadiusFromSpecificAngularMomentum
```

**rotationNormalization** Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for the given node. Specifically, the normalization, $A$, returned is such that $V_{rot} = AJ/M$. Must have the following interface:

```fortran
double precision function myImplementationRotationNormalization(self,node)
  class (darkMatterProfileClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: node
end double precision function myImplementationRotationNormalization
```
17.4. Existing Method Types

**stateStore** Store the state of the object to file. A default implementation exists. If overridden the following interface must be used:

```fortran
subroutine myImplementationStateStore(self, stateFile, fgslStateFile)
  class (darkMatterProfileClass), intent(inout) :: self
  integer, intent(in) :: stateFile
  type (fgsl_file), intent(in) :: fgslStateFile
end subroutine myImplementationStateStore
```

**stateRestore** Restore the state of the object to file. A default implementation exists. If overridden the following interface must be used:

```fortran
subroutine myImplementationStateRestore(self, stateFile, fgslStateFile)
  class (darkMatterProfileClass), intent(inout) :: self
  integer, intent(in) :: stateFile
  type (fgsl_file), intent(in) :: fgslStateFile
end subroutine myImplementationStateRestore
```

**calculationReset** Reset the calculation state of the object. A default implementation exists. If overridden the following interface must be used:

```fortran
subroutine myImplementationCalculationReset(self, thisNode)
  class (darkMatterProfileClass), intent(inout) :: self
  type (treeNode), intent(inout), pointer :: thisNode
end subroutine myImplementationCalculationReset
```

**isFinalizable** Return true if this object can be finalized. A default implementation exists. If overridden the following interface must be used:

```fortran
logical function myImplementationIsFinalizable(self)
  class (darkMatterProfileClass), intent(inout) :: self
end function myImplementationIsFinalizable
```

Existing implementations are:

- **darkMatterProfileIsothermal** Isothermal dark matter halo profiles
- **darkMatterProfileEinasto** “Einasto” dark matter halo profiles
- **darkMatterProfileNFW** Navarro et al. [1997] dark matter halo profiles

**Dark Matter Profile Concentrations**

Additional implementations for dark matter profile concentrations are added using the `darkMatterProfileConcentration` class. The implementation should be placed in a file containing the directive:

```fortran
!# <darkMatterProfileConcentration name="darkMatterProfileConcentrationMyImplementation">
!# <description>A short description of the implementation.</description>
!# </darkMatterProfileConcentration>
```

where `MyImplementation` is an appropriate name for the implementation. This file should be treated as a regular Fortran module, but without the initial `module` and final `end module` lines. That is, it may contain `use` statements and variable declarations prior to the `contains` line, and should contain all functions required by the implementation after that line. Function names should begin with `darkMatterProfileConcentrationMyImplementation`. The file must define a type that extends the
darkMatterProfileConcentrationClass class (or extends another type which is itself an extension of the darkMatterProfileConcentrationClass class), containing any data needed by the implementation along with type-bound functions required by the implementation. The following type-bound functions are required (unless inherited from the parent type):

**concentration** Returns the concentration parameter for the given node. Must have the following interface:

```fortran
double precision function myImplementationConcentration(self,node)
  class(darkMatterProfileConcentrationClass), intent(inout) :: self
  type (treeNode ), intent(inout), pointer :: node
end double precision function myImplementationConcentration
```

darkMatterProfileDefinition Returns a darkMatterProfile object describing the dark matter density profile used to define this concentration. Must have the following interface:

```fortran
class(darkMatterProfileClass) function myImplementationDarkMatterProfileDefinition(self&
  class(darkMatterProfileConcentrationClass), intent(inout) :: self
end class(darkMatterProfileClass) function myImplementationDarkMatterProfileDefinition
```

densityContrastDefinition Returns a virialDensityContrast object describing the virial density contrast used to define this concentration. Must have the following interface:

```fortran
class(virialDensityContrastClass) function myImplementationDensityContrastDefinition(
  class(darkMatterProfileConcentrationClass), intent(inout) :: self
end class(virialDensityContrastClass) function &
& myImplementationDensityContrastDefinition
```

**isFinalizable** Return true if this object can be finalized. A default implementation exists. If overridden the following interface must be used:

```fortran
logical function myImplementationIsFinalizable(self)
  class(darkMatterProfileConcentrationClass), intent(inout) :: self
end logical function myImplementationIsFinalizable
```

Existing implementations are:

darkMatterProfileConcentrationMunozCuartas2011 Dark matter halo concentrations are computed using the algorithm of Muñoz-Cuartas et al. [2011]. See §13.8.3.
darkMatterProfileConcentrationDuttonMaccio2014 Dark matter halo concentrations are computed using the algorithm of Dutton and Macciò [2014]. See §13.8.3.
darkMatterProfileConcentrationNFW1996 Dark matter halo concentrations are computed using the algorithm of Navarro et al. [1996]. See §13.8.3.
darkMatterProfileConcentrationPrada2011 Dark matter halo concentrations are computed using the algorithm of Prada et al. [2011]. See §13.8.3.
darkMatterProfileConcentrationWDM Dark matter halo concentrations are computed using the modifier of Schneider et al. [2012].
17.4. Existing Method Types

darkMatterProfileConcentrationZhao2009 Dark matter halo concentrations are computed using the algorithm of Zhao et al. [2009]. See §13.8.3.

darkMatterProfileConcentrationGao2008 Dark matter halo concentrations are computed using the algorithm of Gao et al. [2008]. See §13.8.3.

Hot Halo Mass Distributions

Additional implementations for hot halo mass distributions are added using the `hotHaloMassDistribution` class. The implementation should be placed in a file containing the directive:

```fortran
!# <hotHaloMassDistribution name="hotHaloMassDistributionMyImplementation">
!# <description>A short description of the implementation.</description>
!# </hotHaloMassDistribution>
```

where `MyImplementation` is an appropriate name for the implementation. This file should be treated as a regular Fortran module, but without the initial `module` and final `end module` lines. That is, it may contain `use` statements and variable declarations prior to the `contains` line, and should contain all functions required by the implementation after that line. Function names should begin with `hotHaloMass-`.

The file must define a type that extends the `hotHaloMassDistributionClass` class (or extends another type which is itself an extension of the `hotHaloMassDistributionClass` class), containing any data needed by the implementation along with type-bound functions required by the implementation. The following type-bound functions are required (unless inherited from the parent type):

- **density** Return the density of the hot halo at the given radius. Must have the following interface:

  ```fortran
  double precision function myImplementationDensity(self,node,radius)
  class (hotHaloMassDistributionClass), intent(inout) :: self
  type (treeNode ), intent(inout), pointer :: node
  double precision , intent(in ) :: radius
  end double precision function myImplementationDensity
  ```

- **densityLogSlope** Return the logarithmic slope of the density of the hot halo at the given radius. Must have the following interface:

  ```fortran
  double precision function myImplementationDensityLogSlope(self,node,radius)
  class (hotHaloMassDistributionClass), intent(inout) :: self
  type (treeNode ), intent(inout), pointer :: node
  double precision , intent(in ) :: radius
  end double precision function myImplementationDensityLogSlope
  ```

- **enclosedMass** Return the mass enclosed in the hot halo at the given radius. Must have the following interface:

  ```fortran
  double precision function myImplementationEnclosedMass(self,node,radius)
  class (hotHaloMassDistributionClass), intent(inout) :: self
  type (treeNode ), intent(inout), pointer :: node
  double precision , intent(in ) :: radius
  end double precision function myImplementationEnclosedMass
  ```

- **radialMoment** Return the density of the hot halo at the given radius. Must have the following interface:

  ```fortran
  double precision function myImplementationRadialMoment(self,node,radius)
  class (hotHaloMassDistributionClass), intent(inout) :: self
  type (treeNode ), intent(inout), pointer :: node
  double precision , intent(in ) :: radius
  end double precision function myImplementationRadialMoment
  ```

737
### Adding New Methods

**double precision function myImplementationRadialMoment(self,node,moment, radius)**

class (hotHaloMassDistributionClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: node
double precision , intent(in ) :: moment, &
& radius
end double precision function myImplementationRadialMoment

**rotationNormalization** Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for node. Specifically, the normalization, \( A \), returned is such that \( V_{rot} = AJ/M \). Must have the following interface:

**double precision function myImplementationRotationNormalization(self,node)**
class(hotHaloMassDistributionClass), intent(inout) :: self
type (treeNode ), intent(inout), pointer :: node
end double precision function myImplementationRotationNormalization

**isFinalizable** Return true if this object can be finalized. A default implementation exists. If overridden the following interface must be used:

**logical function myImplementationIsFinalizable(self)**
class(hotHaloMassDistributionClass), intent(inout) :: self
end logical function myImplementationIsFinalizable

Existing implementations are:

**hotHaloMassDistributionBetaProfile** Provides a \( \beta \)-profile implementation of the hot halo mass distribution class. See §13.16.1.

**hotHaloMassDistributionNull** Provides a null implementation of the hot halo mass distribution class. See §13.16.3.

**hotHaloMassDistributionRicotti2000** Provides an implementation of the hot halo mass distribution class which uses the model of Ricotti and Shull [2000]. See §13.16.2.

### Hot Halo Mass Distributions Core Radii

Additional implementations for hot halo mass distributions core radii are added using the **hotHaloMassDistributionCoreRadius** class. The implementation should be placed in a file containing the directive:

```
!# <hotHaloMassDistributionCoreRadius name="hotHaloMassDistributionCoreRadiusMyImplementation">
!# <description>A short description of the implementation.</description>
!# </hotHaloMassDistributionCoreRadius>
```

where **MyImplementation** is an appropriate name for the implementation. This file should be treated as a regular Fortran module, but without the initial module and final end module lines. That is, it may contain use statements and variable declarations prior to the contains line, and should contain all functions required by the implementation after that line. Function names should begin with hot-HaloMassDistributionCoreRadiusMyImplementation. The file must define a type that extends the **hotHaloMassDistributionCoreRadiusClass** class (or extends another type which is itself an extension of the **hotHaloMassDistributionCoreRadiusClass** class), containing any data needed by the implementation along with type-bound functions required by the implementation. The following type-bound functions are required (unless inherited from the parent type):

**radius** Return the core radius of the hot halo mass distribution. Must have the following interface:
double precision function myImplementationRadius(self,node)
  class (hotHaloMassDistributionCoreRadiusClass), intent(inout) :: self
  type (treeNode ), intent(inout), pointer :: node
end double precision function myImplementationRadius

stateStore Store the state of the object to file. A default implementation exists. If overridden the following interface must be used:

subroutine myImplementationStateStore(self,stateFile,fgslStateFile)
  class (hotHaloMassDistributionCoreRadiusClass), intent(inout) :: self
  integer , intent(in ) :: stateFile
  type (fgsl_file ), intent(in ) :: fgslStateFile
end subroutine myImplementationStateStore

stateRestore Restore the state of the object to file. A default implementation exists. If overridden the following interface must be used:

subroutine myImplementationStateRestore(self,stateFile,fgslStateFile)
  class (hotHaloMassDistributionCoreRadiusClass), intent(inout) :: self
  integer , intent(in ) :: stateFile
  type (fgsl_file ), intent(in ) :: fgslStateFile
end subroutine myImplementationStateRestore

isFinalizable Return true if this object can be finalized. A default implementation exists. If overridden the following interface must be used:

logical function myImplementationIsFinalizable(self)
  class(hotHaloMassDistributionCoreRadiusClass), intent(inout) :: self
end logical function myImplementationIsFinalizable

Existing implementations are:

hotHaloMassDistributionCoreRadiusVirialFraction Provides an implementation of the hot halo mass distribution core radius class which sets the core radius to a fraction of the virial radius. See §13.17.2.

hotHaloMassDistributionCoreRadiusGrowing Provides an implementation of the hot halo mass distribution core radius class in which the core grows as the hot halo content is depleted. See §13.17.1.

Intergalactic Medium State

Additional implementations for intergalactic medium state are added using the intergalacticMediumState class. The implementation should be placed in a file containing the directive:

!# <intergalacticMediumState name="intergalacticMediumStateMyImplementation">
!# <description>A short description of the implementation.</description>
!# </intergalacticMediumState>

where MyImplementation is an appropriate name for the implementation. This file should be treated as a regular Fortran module, but without the initial module and final end module lines. That is, it may contain use statements and variable declarations prior to the contains line, and should contain all functions required by the implementation after that line. Function names should begin with intergalacticMedium-StateMyImplementation. The file must define a type that extends the intergalacticMediumStateClass class (or extends another type which is itself an extension of the intergalacticMediumStateClass class), containing any data needed by the implementation along with type-bound functions required by the implementation. The following type-bound functions are required (unless inherited from the parent type):

739
doublyIonizedHeliumFraction  Return the doubly-ionized fraction of helium in the IGM at the given time. A default implementation exists. If overridden the following interface must be used:

    double precision function myImplementationDoublyIonizedHeliumFraction(self,time) 
    class (intergalacticMediumStateClass), intent(inout) :: self 
    double precision , intent(in ) :: time 
end double precision function myImplementationDoublyIonizedHeliumFraction

electronFraction  Return the electron fraction (relative to hydrogen) in the IGM at the given time. Must have the following interface:

    double precision function myImplementationElectronFraction(self,time) 
    class (intergalacticMediumStateClass), intent(inout) :: self 
    double precision , intent(in ) :: time 
end double precision function myImplementationElectronFraction

electronScatteringOpticalDepth  Return the electron scattering optical depth from the present day back to the given time in the IGM. A default implementation exists. If overridden the following interface must be used:

    double precision function myImplementationElectronScatteringOpticalDepth(self,time,& & assumeFullyIonized) 
    class (intergalacticMediumStateClass), intent(inout) :: self 
    double precision , intent(in ) :: time 
    logical , intent(in ), optional :: & 
    & assumeFullyIonized 
end double precision function myImplementationElectronScatteringOpticalDepth

electronScatteringTime  Return the cosmological time at which the given electron scattering opticalDepth is reached (integrating from the present day) in the IGM. A default implementation exists. If overridden the following interface must be used:

    double precision function myImplementationElectronScatteringTime(self,opticalDepth,& & assumeFullyIonized) 
    class (intergalacticMediumStateClass), intent(inout) :: self 
    double precision , intent(in ) :: & 
    & opticalDepth 
    logical , intent(in ), optional :: & 
    & assumeFullyIonized 
end double precision function myImplementationElectronScatteringTime

neutralHeliumFraction  Return the neutral fraction of helium in the IGM at the given time. Must have the following interface:

    double precision function myImplementationNeutralHeliumFraction(self,time) 
    class (intergalacticMediumStateClass), intent(inout) :: self 
    double precision , intent(in ) :: time 
end double precision function myImplementationNeutralHeliumFraction

neutralHydrogenFraction  Return the neutral fraction of hydrogen in the IGM at the given time. Must have the following interface:

    double precision function myImplementationNeutralHydrogenFraction(self,time) 
    class (intergalacticMediumStateClass), intent(inout) :: self 
    double precision , intent(in ) :: time 
end double precision function myImplementationNeutralHydrogenFraction
singlyIonizedHeliumFraction Return the singly-ionized fraction of helium in the IGM at the given time. Must have the following interface:

```fortran
double precision function myImplementationSinglyIonizedHeliumFraction(self,time)
  class (intergalacticMediumStateClass), intent(inout) :: self
double precision , intent(in ) :: time
end double precision function myImplementationSinglyIonizedHeliumFraction
```

singlyIonizedHydrogenFraction Return the singly-ionized fraction of hydrogen in the IGM at the given time. A default implementation exists. If overridden the following interface must be used:

```fortran
double precision function myImplementationSinglyIonizedHydrogenFraction(self,time)
  class (intergalacticMediumStateClass), intent(inout) :: self
double precision , intent(in ) :: time
end double precision function myImplementationSinglyIonizedHydrogenFraction
```

temperature Return the temperature (in Kelvin) of the IGM at the given time. Must have the following interface:

```fortran
double precision function myImplementationTemperature(self,time)
  class (intergalacticMediumStateClass), intent(inout) :: self
double precision , intent(in ) :: time
end double precision function myImplementationTemperature
```

isFinalizable Return true if this object can be finalized. A default implementation exists. If overridden the following interface must be used:

```fortran
logical function myImplementationIsFinalizable(self)
  class(intergalacticMediumStateClass), intent(inout) :: self
end logical function myImplementationIsFinalizable
```

Existing implementations are:

intergalacticMediumStateSimple The intergalactic medium is assumed to be instantaneously and fully reionized at a fixed redshift, and heated to a fixed temperature. See §13.23.1.

intergalacticMediumStateInternal The intergalactic medium is assumed to be instantaneously and fully reionized at a fixed redshift, and heated to a fixed temperature.

intergalacticMediumStateFile The intergalactic medium state is read from file. See §13.23.3.

intergalacticMediumStateRecFast The intergalactic medium state is computed using RECFAST. See §13.23.2.

Merger Tree Importer

Additional implementations for merger tree importer are added using the mergerTreeImporter class. The implementation should be placed in a file containing the directive:

```markup
!# <mergerTreeImporter name="mergerTreeImporterMyImplementation">
!# <description>A short description of the implementation.</description>
!# </mergerTreeImporter>
```
where *MyImplementation* is an appropriate name for the implementation. This file should be treated as a regular Fortran module, but without the initial *module* and final *end module* lines. That is, it may contain use statements and variable declarations prior to the *contains* line, and should contain all functions required by the implementation after that line. Function names should begin with *mergerTree-ImporterMyImplementation*. The file *must* define a type that extends the *mergerTreeImporterClass* class (or extends another type which is itself an extension of the *mergerTreeImporterClass* class), containing any data needed by the implementation along with type-bound functions required by the implementation. The following type-bound functions are required (unless inherited from the parent type):

**angularMomenta3DAvailable** Return true if angular momenta (vectors) are available. Must have the following interface:

```fortran
logical function myImplementationAngularMomenta3DAvailable(self)
  class(mergerTreeImporterClass), intent(inout) :: self
end logical function myImplementationAngularMomenta3DAvailable
```

**angularMomentaAvailable** Return true if angular momenta (magnitudes) are available. Must have the following interface:

```fortran
logical function myImplementationAngularMomentaAvailable(self)
  class(mergerTreeImporterClass), intent(inout) :: self
end logical function myImplementationAngularMomentaAvailable
```

**angularMomentaIncludeSubhalos** Returns a Boolean specifying whether halo angular momenta (or spins) include the contribution from their subhalos. Must have the following interface:

```fortran
logical function myImplementationAngularMomentaIncludeSubhalos(self)
  class(mergerTreeImporterClass), intent(inout) :: self
end logical function myImplementationAngularMomentaIncludeSubhalos
```

**close** Closes the file. Must have the following interface:

```fortran
subroutine myImplementationClose(self)
  class(mergerTreeImporterClass), intent(inout) :: self
end subroutine myImplementationClose
```

**cubeLength** Returns the length of the simulation cube. Must have the following interface:

```fortran
double precision function myImplementationCubeLength(self, time, status)
  class (mergerTreeImporterClass), intent(inout) :: self
  double precision , intent(in ) :: time
  integer , intent( out), optional :: status
end double precision function myImplementationCubeLength
```

**import** Imports the *i*\(^{th}\) tree. Must have the following interface:

```fortran
subroutine myImplementationImport(self, i, nodes, requireScaleRadii, requireAngularMomenta&\ & , requireAngularMomenta3D, requireSpin, requireSpin3D, requirePositions, &\ & requireParticleCounts, requireVelocityMaxima, requireVelocityDispersions)
  class (mergerTreeImporterClass) , intent(inout) &
  integer , intent(in ) &
end subroutine myImplementationImport
```

17. Adding New Methods
17.4. Existing Method Types

class (nodeData ), allocatable, dimension(:,), intent( out) 
& :: nodes 
logical , intent(in ), optional & 
& :: requireScaleRadii, requireAngularMomenta, requireAngularMomenta3D, requireSpin, & 
& requireSpin3D, requirePositions, requireParticleCounts, requireVelocityMaxima, & 
& requireVelocityDispersions 
end subroutine myImplementationImport

massesIncludeSubhalos Returns a Boolean specifying whether halo masses include the contribution from their subhalos. Must have the following interface:

logical function myImplementationMassesIncludeSubhalos(self)
class(mergerTreeImporterClass), intent(inout) :: self 
end logical function myImplementationMassesIncludeSubhalos

nodeCount Returns the number of nodes in the $i^{th}$ tree. Must have the following interface:

integer(kind=c_size_t) function myImplementationNodeCount(self,i)
class (mergerTreeImporterClass), intent(inout) :: self 
integer , intent(in ) :: i 
end integer(kind=c_size_t) function myImplementationNodeCount

open Opens the file. Must have the following interface:

subroutine myImplementationOpen(self,fileName)
class(mergerTreeImporterClass), intent(inout) :: self 
type (varying_string ), intent(in ) :: fileName 
end subroutine myImplementationOpen

particleCountAvailable Return true if particle counts are available. Must have the following interface:

logical function myImplementationParticleCountAvailable(self)
class(mergerTreeImporterClass), intent(inout) :: self 
end logical function myImplementationParticleCountAvailable

positionsArePeriodic Returns a Boolean integer specifying whether positions are periodic. Must have the following interface:

integer function myImplementationPositionsArePeriodic(self)
class(mergerTreeImporterClass), intent(inout) :: self 
end integer function myImplementationPositionsArePeriodic

positionsAvailable Return true if positions and/or velocities are available. Must have the following interface:

logical function myImplementationPositionsAvailable(self,positions, velocities)
class (mergerTreeImporterClass), intent(inout) :: self 
logical , intent(in ) :: positions, velocities 
end logical function myImplementationPositionsAvailable

scaleRadiiAvailable Return true if scale radii are available. Must have the following interface:

logical function myImplementationScaleRadiiAvailable(self)
class(mergerTreeImporterClass), intent(inout) :: self 
end logical function myImplementationScaleRadiiAvailable
17. Adding New Methods

**spin3DAvailable** Return true if spin (vectors) are available. Must have the following interface:

```fortran
logical function myImplementationSpin3DAvailable(self)
  class(mergerTreeImporterClass), intent(inout) :: self
end logical function myImplementationSpin3DAvailable
```

**spinAvailable** Return true if spin (magnitudes) are available. Must have the following interface:

```fortran
logical function myImplementationSpinAvailable(self)
  class(mergerTreeImporterClass), intent(inout) :: self
end logical function myImplementationSpinAvailable
```

**subhaloTrace** Supplies epochs, positions, and velocities for traced subhalos. Must have the following interface:

```fortran
subroutine myImplementationSubhaloTrace(self,node,time,position, velocity)
  class (mergerTreeImporterClass) , intent(inout) :: self
  class (nodeData ) , intent(in ) :: node
  double precision , dimension(:) , intent( out) :: time
  double precision , dimension(:,:), intent( out) :: position, &
  & velocity
end subroutine myImplementationSubhaloTrace
```

**subhaloTraceCount** Returns the length of a node’s subhalo trace. Must have the following interface:

```fortran
integer(kind=c_size_t) function myImplementationSubhaloTraceCount(self,node)
  class(mergerTreeImporterClass), intent(inout) :: self
  class(nodeData ), intent(in ) :: node
end integer(kind=c_size_t) function myImplementationSubhaloTraceCount
```

**treeCount** Returns a count of the number of trees available. Must have the following interface:

```fortran
integer(kind=c_size_t) function myImplementationTreeCount(self)
  class(mergerTreeImporterClass), intent(inout) :: self
end integer(kind=c_size_t) function myImplementationTreeCount
```

**treeIndex** Returns the index of the $i^{th}$ tree. Must have the following interface:

```fortran
integer(kind=kind_int8) function myImplementationTreeIndex(self,i)
  class (mergerTreeImporterClass), intent(inout) :: self
  integer , intent(in ) :: i
end integer(kind=kind_int8) function myImplementationTreeIndex
```

**treeWeight** Returns the weight to assign to the $i^{th}$ tree. Must have the following interface:

```fortran
double precision function myImplementationTreeWeight(self,i)
  class (mergerTreeImporterClass), intent(inout) :: self
  integer , intent(in ) :: i
end double precision function myImplementationTreeWeight
```

**treesAreSelfContained** Returns a Boolean integer specifying whether trees are self-contained. Must have the following interface:

```fortran
integer function myImplementationTreesAreSelfContained(self)
  class(mergerTreeImporterClass), intent(inout) :: self
end integer function myImplementationTreesAreSelfContained
```
17.4. Existing Method Types

treesHaveSubhalos Returns a Boolean integer specifying whether or not the trees have subhalos. Must have the following interface:

```
integer function myImplementationTreesHaveSubhalos(self)
class(mergerTreeImporterClass), intent(inout) :: self
end integer function myImplementationTreesHaveSubhalos
```

velocitiesIncludeHubbleFlow Returns a Boolean integer specifying whether velocities include the Hubble flow. Must have the following interface:

```
integer function myImplementationVelocitiesIncludeHubbleFlow(self)
class(mergerTreeImporterClass), intent(inout) :: self
end integer function myImplementationVelocitiesIncludeHubbleFlow
```

velocityDispersionAvailable Return true if halo velocity dispersions are available. Must have the following interface:

```
logical function myImplementationVelocityDispersionAvailable(self)
class(mergerTreeImporterClass), intent(inout) :: self
end logical function myImplementationVelocityDispersionAvailable
```

velocityMaximumAvailable Return true if rotation curve velocity maxima are available. Must have the following interface:

```
logical function myImplementationVelocityMaximumAvailable(self)
class(mergerTreeImporterClass), intent(inout) :: self
end logical function myImplementationVelocityMaximumAvailable
```

isFinalizable Return true if this object can be finalized. A default implementation exists. If overridden the following interface must be used:

```
logical function myImplementationIsFinalizable(self)
class(mergerTreeImporterClass), intent(inout) :: self
end logical function myImplementationIsFinalizable
```

Existing implementations are:

mergerTreeImporterSussing Importer for “Sussing Merger Trees” format merger tree files [Srisawat et al., 2013]. See §13.30.2.

mergerTreeImporterGalacticus Importer for GALACTICUS format merger tree files. See §13.30.1.

Satellite Merging Timescales

Additional implementations for satellite merging timescales are added using the satelliteMergingTimescales class. The implementation should be placed in a file containing the directive:

```
!# <satelliteMergingTimescales name="satelliteMergingTimescalesMyImplementation">
!# <description>A short description of the implementation.</description>
!# </satelliteMergingTimescales>
```

where MyImplementation is an appropriate name for the implementation. This file should be treated as a regular Fortran module, but without the initial module and final end module lines. That is, it may contain use statements and variable declarations prior to the contains line, and should contain all functions
required by the implementation after that line. Function names should begin with \texttt{satelliteMerging-TimescalesMyImplementation}. The file \textit{must} define a type that extends the \texttt{satelliteMergingTimescalesClass} class (or extends another type which is itself an extension of the \texttt{satelliteMergingTimescalesClass} class), containing any data needed by the implementation along with type-bound functions required by the implementation. The following type-bound functions are required (unless inherited from the parent type):

\textbf{stateRestore} Restore the state of the object from file. A default implementation exists. If overridden the following interface must be used:

\begin{verbatim}
    subroutine myImplementationStateRestore(self, stateFile, fgslStateFile)
        class (satelliteMergingTimescalesClass), intent(inout) :: self
        integer , intent(in ) :: stateFile
        type (fgsl_file ), intent(in ) :: fgslStateFile
    end subroutine myImplementationStateRestore
\end{verbatim}

\textbf{stateSnapshot} Stores a snapshot of the object state. A default implementation exists. If overridden the following interface must be used:

\begin{verbatim}
    subroutine myImplementationStateSnapshot(self)
        class(satelliteMergingTimescalesClass), intent(inout) :: self
    end subroutine myImplementationStateSnapshot
\end{verbatim}

\textbf{stateStore} Store the state of the object to file. A default implementation exists. If overridden the following interface must be used:

\begin{verbatim}
    subroutine myImplementationStateStore(self, stateFile, fgslStateFile)
        class (satelliteMergingTimescalesClass), intent(inout) :: self
        integer , intent(in ) :: stateFile
        type (fgsl_file ), intent(in ) :: fgslStateFile
    end subroutine myImplementationStateStore
\end{verbatim}

\textbf{timeUntilMerging} Return the time (in Gyr) until the satellite will merge with its host given the current orbit. Must have the following interface:

\begin{verbatim}
    double precision function myImplementationTimeUntilMerging(self, thisNode, thisOrbit)
        class(satelliteMergingTimescalesClass), intent(inout) :: self
        type (treeNode ), intent(inout), pointer :: thisNode
        type (keplerOrbit ), intent(inout) :: thisOrbit
    end double precision function myImplementationTimeUntilMerging
\end{verbatim}

\textbf{isFinalizable} Return true if this object can be finalized. A default implementation exists. If overridden the following interface must be used:

\begin{verbatim}
    logical function myImplementationIsFinalizable(self)
        class(satelliteMergingTimescalesClass), intent(inout) :: self
    end logical function myImplementationIsFinalizable
\end{verbatim}

Existing implementations are:

\textbf{satelliteMergingTimescalesPreset} This method assumes that merging times have been preset for every node (or, at least, every node which becomes a satellite). It therefore simply returns the preset merging time. See §13.44.1.
17.4. Existing Method Types

satelliteMergingTimescalesVillalobos2013 Computes the merging timescale using the method of Villalobos et al. [2013] to modify another merging timescale method. See §13.44.1.

satelliteMergingTimescalesWetzelWhite2010 Computes the merging timescale using the method of Wetzel and White [2010]. See §13.44.1.

satelliteMergingTimescalesInfinite Returns an infinite timescale for merging. See §13.44.1.

satelliteMergingTimescalesJiang2008 Computes the merging timescale using the method of Jiang et al. [2008]. See §13.44.1.


satelliteMergingTimescalesNull Returns a zero timescale for merging. See §13.44.1.


Spectra Postprocessor

Additional implementations for spectra postprocessor are added using the spectraPostprocessor class. The implementation should be placed in a file containing the directive:

```
#!/ <spectraPostprocessor name="spectraPostprocessorMyImplementation">
#!/ <description>A short description of the implementation.<description>
#!/ </spectraPostprocessor>
```

where MyImplementation is an appropriate name for the implementation. This file should be treated as a regular Fortran module, but without the initial module and final end module lines. That is, it may contain use statements and variable declarations prior to the contains line, and should contain all functions required by the implementation after that line. Function names should begin with spectraPostprocessor-MyImplementation. The file must define a type that extends the spectraPostprocessorClass class (or extends another type which is itself an extension of the spectraPostprocessorClass class), containing any data needed by the implementation along with type-bound functions required by the implementation. The following type-bound functions are required (unless inherited from the parent type):

apply Apply postprocessing to a spectrum. Must have the following interface:

```
subroutine myImplementationApply(self,wavelength, age, redshift,modifier)
   class (spectraPostprocessorClass), intent(inout) :: self
   double precision , intent(in ) :: wavelength, age, &
   redshift
   double precision , intent(inout) :: modifier
end subroutine myImplementationApply
```

isFinalizable Return true if this object can be finalized. A default implementation exists. If overridden the following interface must be used:

```
logical function myImplementationIsFinalizable(self)
   class(spectraPostprocessorClass), intent(inout) :: self
end logical function myImplementationIsFinalizable
```
17. Adding New Methods

Existing implementations are:

**spectraPostprocessorIdentity** Performs an identity postprocessing of spectra. See §13.42.6.

**spectraPostprocessorRecent** Retains only recent stellar populations. See §13.42.5.

**spectraPostprocessorLycSuppress** Suppress the Lyman continuum in stellar populations. See §13.42.4.

**spectraPostprocessorInoue2014** Apply the Inoue et al. [2014] calculation of the attenuation of spectra by the intergalactic medium. See §13.42.1.

**spectraPostprocessorMadau1995** Apply the Madau [1995] calculation of the attenuation of spectra by the intergalactic medium. See §13.42.3.

**spectraPostprocessorMeiksin2006** Apply the Meiksin [2006] calculation of the attenuation of spectra by the intergalactic medium. See §13.42.2.

### Virial Density Contrasts

Additional implementations for virial density contrasts are added using the `virialDensityContrast` class. The implementation should be placed in a file containing the directive:

```fortran
#!/ <virialDensityContrast name="virialDensityContrastMyImplementation">
#!/ <description>A short description of the implementation.</description>
#!/ </virialDensityContrast>
```

where `MyImplementation` is an appropriate name for the implementation. This file should be treated as a regular Fortran module, but without the initial `module` and final `end module` lines. That is, it may contain `use` statements and variable declarations prior to the `contains` line, and should contain all functions required by the implementation after that line. Function names should begin with `virialDensityContrastMyImplementation`. The file must define a type that extends the `virialDensityContrastClass` class (or extends another type which is itself an extension of the `virialDensityContrastClass` class), containing any data needed by the implementation along with type-bound functions required by the implementation. The following type-bound functions are required (unless inherited from the parent type):

**densityContrast** Returns the virial density contrast at the given epoch. Must have the following interface:

```fortran
double precision function myImplementationDensityContrast(self,time , &
& expansionFactor,collapsing)
  class [(virialDensityContrastClass), intent(inout)] :: self
  double precision , intent(in ), optional :: time, &
& expansionFactor
  logical , intent(in ), optional :: collapsing
end double precision function myImplementationDensityContrast
```

**densityContrastRateOfChange** Returns the rate of change of virial density contrast at the given epoch. Must have the following interface:

```fortran
double precision function myImplementationDensityContrastRateOfChange(self,time , &
& expansionFactor,collapsing)
  class [(virialDensityContrastClass), intent(inout)] :: self
  double precision , intent(in ), optional :: time, &
& expansionFactor
  logical , intent(in ), optional :: collapsing
end double precision function myImplementationDensityContrastRateOfChange
```
**17.4. Existing Method Types**

**stateStore** Store the state of the object to file. A default implementation exists. If overridden the following interface must be used:

```fortran
subroutine myImplementationStateStore(self, stateFile, fgslStateFile)
  class (virialDensityContrastClass), intent(inout) :: self
  integer , intent(in ) :: stateFile
  type (fgsl_file ), intent(in ) :: fgslStateFile
end subroutine myImplementationStateStore
```

**stateRestore** Restore the state of the object to file. A default implementation exists. If overridden the following interface must be used:

```fortran
subroutine myImplementationStateRestore(self, stateFile, fgslStateFile)
  class (virialDensityContrastClass), intent(inout) :: self
  integer , intent(in ) :: stateFile
  type (fgsl_file ), intent(in ) :: fgslStateFile
end subroutine myImplementationStateRestore
```

**isFinalizable** Return true if this object can be finalized. A default implementation exists. If overridden the following interface must be used:

```fortran
logical function myImplementationIsFinalizable(self)
  class(virialDensityContrastClass), intent(inout) :: self
end logical function myImplementationIsFinalizable
```

Existing implementations are:

- **virialDensityContrastSphericalCollapseMatterLambda** Dark matter halo virial density contrasts based on the spherical collapse in a matter plus cosmological constant universe.
- **virialDensityContrastFriendsOfFriends** Dark matter halo virial density contrasts based on the friends-of-friends algorithm linking length.
- **virialDensityContrastFixed** Fixed dark matter halo virial density contrasts.
- **virialDensityContrastSphericalCollapseMatterDE** Dark matter halo virial density contrasts based on the spherical collapse in a matter plus dark energy universe.

**Accretion Disks**

Additional methods for accretion disk properties can be added using the `accretionDisksMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the **Shakura-Sunyaev** method is described by a directive:

```
!# <accretionDisksMethod>
!# <unitName>Accretion_Disks_Shakura_Sunyaev_Initialize</unitName>
!# </accretionDisksMethod>
```

Here, **Accretion_Disks_Shakura_Sunyaev_Initialize** is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:
17. Adding New Methods


implicit none

type(varying_string), intent(in) :: accretionDisksMethod
procedure(), pointer, intent(inout) :: Accretion_Disk_Radiative_Efficiency_Get, Black_Hole_Spin_Up_Rate_Get, Accretion_Disk_Jet_Power_Get

if (accretionDisksMethod == 'myMethod') then
    Accretion_Disk_Radiative_Efficiency_Get => My_Accretion_Disk_Radiative_Efficiency_Get
    Black_Hole_Spin_Up_Rate_Get => My_Black_Hole_Spin_Up_Rate_Get
end if

return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the accretionDisksMethod input parameter. The procedure pointers Accretion_Disk_Radiative_Efficiency_Get, Black_Hole_Spin_Up_Rate_Get and Accretion_Disk_Jet_Power_Get must be set to point to functions which return the radiative efficiency, black hole spin up rate and jet power for the accretion disk respectively as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.). The radiative efficiency function must have the form:

double precision function My_Accretion_Disk_Radiative_Efficiency_Get(thisNode, massAccretionRate)

implicit none

type(treeNode), intent(inout), pointer :: thisNode

double precision, intent(in) :: massAccretionRate
.
.
return
end function My_Accretion_Disk_Radiative_Efficiency_Get

The function must return the radiative efficiency for the accretion disk in thisNode. The black hole spin function must have the form:

double precision function My_Black_Hole_Spin_Up_Rate_Get(thisNode, massAccretionRate)

implicit none

type(treeNode), intent(inout), pointer :: thisNode

double precision, intent(in) :: massAccretionRate
.
.
return
end function My_Black_Hole_Spin_Up_Rate_Get

The function must return the spin-up rate for the black hole in thisNode given the massAccretionRate. The jet power function must have the form:

double precision function My_Accretion_Disk_Jet_Power_Get(thisNode, massAccretionRate)

implicit none

type(treeNode), intent(inout), pointer :: thisNode

double precision, intent(in) :: massAccretionRate
.
.
return
end function My_Accretion_Disk_Jet_Power_Get

The function must return the jet power for the accretion disk in thisNode given the massAccretionRate.
17.4. Existing Method Types

The function must return (in units of $M_\odot (\text{km/s})^2 \text{Gyr}^{-1}$) the jet power for the black hole/accretion disk system in thisNode given the massAccretionRate.

Currently defined accretion disk methods are:

**Shakura-Sunyaev** Computes the properties of a thin, radiatively efficiency accretion disk.

**ADAF** Computes the properties of an ADAF using the model of Benson and Babul [2009].

**Switched** Select either Shakura-Sunyaev or ADAF accretion disks based on the accretion rate:

$$\dot{m}_{\text{minimum}} < \dot{M}_{\text{Eddington}} < \dot{m}_{\text{maximum}} \rightarrow \text{Shakura-Sunyaev}$$

otherwise $\rightarrow$ ADAF, \hspace{1cm} (17.2)

where $\dot{m}_{\text{minimum}}$=accretionRateThinDiskMinimum and $\dot{m}_{\text{maximum}}$=accretionRateThinDiskMaximum are input parameters.

**EddingtonLimited** Assumes no specific disk structure, instead setting the radiative efficiency to a fixed number and the jet power to a fixed fraction of the Eddington luminosity.

**Accretion Onto Halos**

Additional methods for accretion of baryons onto halos can be added using the accretionHalosMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the **simple** method is described by a directive:

```plaintext
!# <accretionHalosMethod>
!# <unitName>Accretion_Halos_Simple.Initialize</unitName>
!# </accretionHalosMethod>
```

Here, **Accretion_Halos_Simple.Initialize** is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
subroutine Method.Initialize(accretionHalosMethod,Halo_Baryonic_Accretion_Rate_Get,Halo_Baryonic_Accreted_Mass_Get, &
Halo_Baryonic_Failed_Accretion_Rate_Get,Halo_Baryonic_Failed_Accreted_Mass_Get, &
Halo_Baryonic_Accretion_Rate_Abundances_Get,Halo_Baryonic_Accreted_Abundances_Get, &
Halo_Baryonic_Accretion_Rate_Chemicals_Get,Halo_Baryonic_Accreted_Chemicals_Get)
```

```
implicit none

type(varying_string), intent(in) :: accretionHalosMethod

procedure(), pointer, intent(inout) :: Halo_Baryonic_Accretion_Rate_Get,Halo_Baryonic_Accreted_Mass_Get, &
Halo_Baryonic_Failed_Accretion_Rate_Get,Halo_Baryonic_Failed_Accreted_Mass_Get, &
Halo_Baryonic_Accretion_Rate_Abundances_Get,Halo_Baryonic_Accreted_Abundances_Get, &
Halo_Baryonic_Accretion_Rate_Chemicals_Get,Halo_Baryonic_Accreted_Chemicals_Get
```

```plaintext
if (accretionHalosMethod == 'myMethod') then
  Halo_Baryonic_Accretion_Rate_Get => My_Accretion_Rate_Get
  Halo_Baryonic_Accreted_Mass_Get => My_Accreted_Mass_Get
  Halo_Baryonic_Failed_Accretion_Rate_Get => My_Failed_Accretion_Rat_Get
  Halo_Baryonic_Failed_Accreted_Mass_Get => My_Failed_Accreted_Mass_Get
```

751
17. Adding New Methods

Halo_Baryonic_Accretion_Rate_Abundances_Get => My_Accretion_Rate_Abundances_Get
Halo_Baryonic_Accreted_Abundances_Get => My_Accreted_Abundances_Get
Halo_Baryonic_Accretion_Rate_Chemicals_Get => My_Accretion_Rate_Chemicals_Get
Halo_Baryonic_Accreted_Chemicals_Get => My_Accreted_Chemicals_Get
end if
return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the accretionHalosMethod input parameter. The procedure pointers Halo_Baryonic_Accretion_Rate_Get, Halo_Baryonic_Accreted_Mass_Get, Halo_Baryonic_Failed_Accretion_Rate_Get and Halo_Baryonic_Failed_Accreted_Mass_Get must be set to point to functions which return accretion rate, total accreted mass (assuming no progenitors), failed accretion rate and total failed accreted mass (assuming no progenitors) respectively as described below. The procedure pointers Halo_Baryonic_Accretion_Rate_Abundances_Get, Halo_Baryonic_Accreted_Abundances_Get, Halo_Baryonic_Accretion_Rate_Chemicals_Get, Halo_Baryonic_Accreted_Chemicals_Get must be set to point to functions which return the accretion rates nad masses (assuming no progenitors) of heavy element abundances and chemicals respectively. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The mass functions must have the form:

double precision function My_Accretion_Get(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  ...
  return
end function My_Accretion_Get

In the case of the accretion rate functions, the function must return the accretion rate of baryons from the IGM onto thisNode in $M_\odot \text{ Gyr}^{-1}$. For total accreted mass functions, the total mass of baryons (in $M_\odot$) accreted onto thisNode should be returned under the assumption that thisNode formed instataneously with no progenitors. The “failed” accretion refers to mass which would have been accreted onto the halo if it simply traced the growth of overall mass. That is:

$$M_{\text{failed}} = \frac{\Omega_b}{\Omega_M} \dot{M} - \dot{M}_{\text{accreted}}, \quad (17.3)$$

where $\dot{M}$ is the growth rate of total halo mass and $\dot{M}_{\text{accreted}}$ is the accretion rate of baryons onto the halo. If desired, this failed mass can be transferred back into the accreted component once the halo is deemed able to accrete, by simply adjusting the accretion rates returned appropriately.

For abundances and chemicals, the subroutines should have the form:

subroutine My_Abundances_Get(thisNode, accretionAbundances)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  type(abundancesStructure), intent(inout) :: accretionAbundances
  ...
  return
end subroutine My_Abundances_Get

and
subroutine My_Chemicals_Get(thisNode, accretionChemicals)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  type(chemicalAbundancesStructure), intent(inout) :: accretionChemicals

  return
end subroutine My_Chemicals_Get

respectively.

Currently defined accretion disk methods are:

css
t

assumes that halos accrete all available baryons if they have virial velocities above \( \text{reionizationSuppressionVelocity} \) or exist prior to redshift \( \text{reionizationSuppressionRedshift} \). This is a simple model of the effects of reionization on gas accretion. In halos which cannot accrete, accretion is placed into the failed mode. In halos which can accrete, any gas in the failed reservoir is returned to the accreted channel on a timescale of \( \dot{M}/\dot{M} \). Abundances are computed assuming a pristine IGM (i.e. abundances are always zero) and chemicals are computed using the chemical state functions (see §17.4.1).

null

assumes no accretion onto halos.

Analysis

Additional methods for on-the-fly analysis of merger trees can be added using the mergerTreeAnalysisTask directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the mass distributions method is described by a directive:

```
! <mergerTreeAnalysisTask>
! <unitName>Galacticus_Output_Analysis_Mass_Dpndnt_Sz_Dstrbtins</unitName>
! </mergerTreeAnalysisTask>
```

Here, \( \text{Galacticus_Output_Analysis_Mass_Dpndnt_Sz_Dstrbtins} \) is the name of a subroutine which will be called to perform the analysis. The analysis subroutine must have the following form:

```
subroutine Tree_Analyze(thisTree, thisNode, iOutput, mergerTreeAnalyses)
  implicit none
  type (mergerTree ), intent(in ) :: thisTree
  type (treeNode ), intent(inout), pointer :: thisNode
  integer , intent(in ) :: iOutput
  type (varying_string), intent(in ), dimension(: ) :: mergerTreeAnalyses

  return
end subroutine Method_Initialize
```

This function will be called once for each node in each tree being output. The function is passed the merger tree object as thisTree, along with a pointer to the node to be analyzed as thisNode. Additionally, the current output number is passed as iOutput. Finally, a list of analyses that were requested (by user input) to be performed is given in the mergerTreeAnalyses array. The function should check if one or more of the entries in mergerTreeAnalyses correspond to analyses that it performs. (Note that this need only be done on the first call to this function—the values of mergerTreeAnalyses will not change between calls.) If an analysis is matched in this way it should be performed. Typically, an analysis
17. Adding New Methods

function might accumulate the results of analysis and then finalize and output them prior to completion of the GALACTICUS model through the use of a hdfPreCloseTask (see §17.4.3).

Currently defined merger tree analysis methods are:

**mass functions** Constructs mass functions for a variety of different surveys. Currently supported analysis names are: sdssStellarMassFunctionZ0.07, alfalfaHiMassFunctionZ0.00, primusStellarMassFunctionZ0.100, primusStellarMassFunctionZ0.250, primusStellarMassFunctionZ0.350, primusStellarMassFunctionZ0.450, primusStellarMassFunctionZ0.575, primusStellarMassFunctionZ0.725, and primusStellarMassFunctionZ0.900. Each such analysis is defined internally to this module through the appropriate redshift, binning, random and systematic errors, and arbitrary mapping of component masses into observed masses. The covariance matrix of the mass function is also computed (see §10.5.3 for details).

**mass-dependent size distributions** Constructs a two-dimensional histogram of galaxy sizes in bins of stellar mass. Currently supported analyses are sdssSizeFunctionZ0.07. Each such analysis is defined internally to this module through the appropriate redshift, binning, random and systematic errors, and arbitrary mapping of component masses/radii into observed masses/radii. The covariance matrix of the size function is also computed (see §10.5.3 for details).

**Atomic Collisional Ionization Rates**

Additional methods for atomic collisional ionization rate calculations can be added using the atomicCollisionalIonizationMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Verner method is described by a directive:

```
!# <atomicCollisionalIonizationMethod>
!#   <unitName>Atomic_Rate_Ionization_Collisional_Verner_Initialize</unitName>
!# </atomicCollisionalIonizationMethod>
```

Here, Atomic_Rate_Ionization_Collisional_Verner_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(atomicCollisionalIonizationMethod, Atomic_Rate_Ionization_Collisional_Get)
  implicit none
  type(varying_string), intent(in) :: atomicCollisionalIonizationMethod
  procedure(), pointer, intent(inout) :: Atomic_Rate_Ionization_Collisional_Get

  if (atomicCollisionalIonizationMethod == 'myMethod') Atomic_Rate_Ionization_Collisional_Get => My_Method_Get_Procedure
  return
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the atomicCollisionalIonizationMethod input parameter. The procedure pointer Atomic_Rate_Ionization_Collisional_Get must be set to point to a function which returns the rate coefficient of atomic collisional ionization under given physical conditions. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The collisional ionization rate function must have the form:

```fortran
double precision function My_Method_Get_Procedure(atomicNumber, ionizationState, temperature)
  implicit none
  integer, intent(in) :: atomicNumber, ionizationState
  double precision, intent(in) :: temperature
```

754
17.4. Existing Method Types

The function must return the collisional ionization rate coefficient (in units of cm$^3$ s$^{-1}$) of ions of the given atomicNumber, ionizationState (where ionization state is the atomic number plus 1 minus the number of electrons) and temperature. Currently defined collisional ionization rate methods are:

**Verner** Computes the rate coefficient of direct collisional ionization by use of the fits from Voronov (1997; Version 2, March 24, 1997).

**Atomic Photoionization Cross-Sections**

Additional methods for atomic photoionization cross-section calculations can be added using the `atomicPhotoIonizationMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the **Verner** method is described by a directive:

```!
<atomicPhotoIonizationMethod>
<unitName>Atomic_Cross_Section_Ionization_Photo_Verner_Initialize</unitName>
</atomicPhotoIonizationMethod>
```

Here, `Atomic_Cross_Section_Ionization_Photo_Verner_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(atomicPhotoIonizationMethod, Atomic_Cross_Section_Ionization_Photo_Get)
  implicit none
  type(varying_string), intent(in) :: atomicPhotoIonizationMethod
  procedure(), pointer, intent(inout) :: Atomic_Cross_Section_Ionization_Photo_Get

  if (atomicPhotoIonizationMethod == 'myMethod') Atomic_Cross_Section_Ionization_Photo_Get => My_Method_Get_Procedure
  return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `atomicPhotoIonizationMethod` input parameter. The procedure pointer `Atomic_Cross_Section_Ionization_Photo_Get` must be set to point to a function which returns the cross-section (in units of cm$^2$) for photoionization. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The cross-section function must have the form:

```fortran
double precision function My_Method_Get_Procedure(atomicNumber, ionizationState, shellNumber, wavelength)
  implicit none
  integer, intent(in) :: atomicNumber, ionizationState, shellNumber
  double precision, intent(in) :: wavelength

  .
  .
  return
end function My_Method_Get_Procedure
```

The function must return the cross-section for photoionization (in units of cm$^2$) of electrons in the specified shellNumber for ions of the given atomicNumber and ionizationState (where ionization state
is the atomic number plus 1 minus the number of electrons) at the specified wavelength (given in units of Å).

Currently defined photoionization cross-section methods are:

**Verner** Computes the cross-sections by use of the fits from Verner et al. (1996; Version 2, March 25, 1996).

### Atomic Radiative Recombination Rates

Additional methods for atomic radiative recombination rate calculations can be added using the atomicRadiativeRecombinationMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Verner method is described by a directive:

```
!# <atomicRadiativeRecombinationMethod>
!# <unitName>Atomic_Rate_Recombination_Radiative_Verner.Initialize</unitName>
!# </atomicRadiativeRecombinationMethod>
```

Here, Atomic_Rate_Recombination_Radiative_Verner.Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(atomicRadiativeRecombinationMethod, Atomic_Rate_Recombination_Radiative_Get)
    implicit none
    type(varying_string), intent(in) :: atomicRadiativeRecombinationMethod
    procedure(), pointer, intent(inout) :: Atomic_Rate_Recombination_Radiative_Get

    if (atomicRadiativeRecombinationMethod == 'myMethod')
        Atomic_Rate_Recombination_Radiative_Get => My_Method_Get_Procedure
    return
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the atomicRadiativeRecombinationMethod input parameter. The procedure pointer Atomic_Rate_Recombination_Radiative_Get must be set to point to a function which returns the rate coefficient of atomic radiative recombination under given physical conditions. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The radiative recombination rate function must have the form:

```fortran
double precision function My_Method_Get_Procedure(atomicNumber, ionizationState, temperature)
    implicit none
    integer, intent(in) :: atomicNumber, ionizationState
    double precision, intent(in) :: temperature

    return
end function My_Method_Get_Procedure
```

The function must return the radiative recombination rate coefficient (in units of cm$^3$ s$^{-1}$) to ions of the given atomicNumber, ionizationState (where ionization state is the atomic number plus 1 minus the number of electrons) and temperature.

Currently defined radiative recombination rate methods are:

**Verner** Computes the rate coefficient of radiative recombination using the compilation of results from Dima Verner as originally encapsulated in rrfit.f.
Bar Instabilities

Additional methods for bar instabilities in disks can be added using the `barInstabilityMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the ELN method is described by a directive:

```bash
!# <barInstabilityMethod>
!# <unitName>Galactic_Dynamics_Bar_Instabilities_ELN_Initialize</unitName>
!# </barInstabilityMethod>
```

Here, `Galactic_Dynamics_Bar_Instabilities_ELN_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
subroutine Method_Initialize(barInstabilityMethod,Bar_Instability_Timescale_Get)
  implicit none
  type(varying_string), intent(in) :: barInstabilityMethod
  procedure(), pointer, intent(inout) :: Bar_Instability_Timescale_Get
  if (barInstabilityMethod == 'myMethod') then
    Bar_Instability_Timescale_Get => My_Bar_Instability_Timescale_Get
  end if
  return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `barInstabilityMethod` input parameter. The procedure pointer `Bar_Instability_Timescale_Get` must be set to point to a function which returns the timescale on which the bar instability depletes material from the disk to the pseudo-bulge. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The bar instability timescale function must have the form:

```plaintext
subroutine My_Bar_Instability_Timescale(thisNode,barInstabilityTimeScale,barInstabilityExternalDrivingSpecificTorque)
  implicit none
  type (treeNode), intent(inout), pointer :: thisNode
  double precision , intent( out) :: barInstabilityTimeScale,barInstabilityExternalDrivingSpecificTorque
  return
end subroutine My_Bar_Instability_Timescale
```

The function should compute and return, in `barInstabilityTimeScale` the timescale (in Gyr) for the bar instability in the disk in `thisNode` to transfer material from the disk to the pseudo-bulge. If no instability is present, a negative timescale should be returned. Additionally, any specific torque external to the galaxy driving the instability should be returned in `barInstabilityExternalDrivingSpecificTorque`.

Currently defined bar instability methods are:

null A null method in which disks are never bar unstable;

ELN The bar instability is determined using the algorithm of Efstathiou et al. [1982].

ELN+tidal The bar instability is determined using the algorithm of Efstathiou et al. [1982] with an additional term to account for an external tidal field.
17. Adding New Methods

**Black Hole Binaries: Initial Separation**

Additional methods for black hole binary initial separation calculations can be added using the `blackHoleBinaryInitialRadiiMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `spheroidRadiusFraction` method is described by a directive:

```
!# <blackHoleBinaryInitialRadiiMethod>
!# <unitName>Black_Hole_Binary_Initial_Radii_Spheroid_SizeInitialize</unitName>
!# </blackHoleBinaryInitialRadiiMethod>
```

Here, `Black_Hole_Binary_Initial_Radii_Spheroid_SizeInitialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(blackHoleBinaryInitialRadiiMethod, Black_Hole_Binary_Initial_Radius_Get)
  implicit none
  type(varying_string), intent(in) :: blackHoleBinaryInitialRadiiMethod
  procedure(), pointer, intent(inout) :: Black_Hole_Binary_Initial_Radius_Get
  if (blackHoleBinaryInitialRadiiMethod == 'myMethod') Black_Hole_Binary_Initial_Radius_Get => My_Method_Get
  return
end subroutine Method_Initialize
```

where `
myMethod
` is the name of this method as will be specified by the `blackHoleBinaryInitialRadiiMethod` input parameter. The procedure pointer `Black_Hole_Binary_Initial_Radius_Get` must be set to point to a function which returns the initial separation of a just-formed black hole binary. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The initial separation function must have the form:

```fortran
double precision function My_Method_Get(thisNode, hostNode)
  implicit none
  type(treeNode), intent(inout) :: thisNode, hostNode
  .
  .
  return
end subroutine My_Method_Get
```

The function must return the initial separation (in Mpc) of the active black hole in `thisNode` as it merges into `hostNode`.

Currently defined black hole binary initial separation methods are:

- **spheroidRadiusFraction** Assumes that the initial separation is equal to a fraction of the larger of the spheroid scale radii in `thisNode` and `hostNode`.

- **Volonteri2003** Assumes that the initial separation follows the relationship described in Volonteri et al. [2003] following the black hole masses in `thisNode` and `hostNode`.

- **tidalRadius** Solves the radius at which the satellite galaxy is stripped of its stars, and assume only the central black hole remains, at that specific radius. This uses the masses in `thisNode` and `hostNode`. 

758
Black Hole Binaries: Separation Growth Rate

Additional methods for black hole binary separation growth rate calculations can be added using the `blackHoleBinarySeparationGrowthRateMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `Volonteri2003` method is described by a directive:

```
!# <blackHoleBinarySeparationGrowthRateMethod>
!# <unitName>Black_Hole_Binary_Separation_Growth_Rate_Standard_Init</unitName>
!# </blackHoleBinarySeparationGrowthRateMethod>
```

Currently defined black hole binary separation growth rate methods are:

- **null** Assures that the initial separation stays constant.
- **standard** Assumes that the separation growth rate follows Volonteri et al. [2003] following the black hole masses in `thisNode`. Although it innovates as it encompasses all three influences: Dynamical Friction, Hardening due to stars, and finally due to Gravitational Wave expulsion. Dynamical friction here occurs until a certain hardening separation is reached, it then is replaced by the (faster) three-body interactions with stars.

Black Hole Binaries: Recoil Velocity

Additional methods for the recoil velocity of a binary black hole can be added using the `blackHoleBinaryRecoilVelocityMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `Standard` method is described by a directive:

```
!# <blackHoleBinaryRecoilVelocityMethod>
!# <unitName>Black_Hole_Binary_Recoil_Velocity_Standard_Initialize</unitName>
!# </blackHoleBinaryRecoilVelocityMethod>
```

Currently defined black hole binary recoil velocity methods are:

- **null** Assures that there is zero recoil velocity.
- **Campanelli2008** Assures that the recoil velocity follows Campanelli et al. [2007], utilizing the black hole masses and spins in `thisNode`. For now it does not take the direction of the spin into account, and assumes a zero perpendicular velocity.

Black Hole Binaries: Mergers

Additional methods for black hole binary merger calculations can be added using the `blackHoleBinaryMergersMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `Rezzolla2008` method is described by a directive:

```
!# <blackHoleBinaryMergersMethod>
!# <unitName>Black_Hole_Binary_Merger_Initialize</unitName>
!# </blackHoleBinaryMergersMethod>
```

Here, **Black_Hole_Binary_Merger_Initialize** is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:
17. Adding New Methods

subroutine Method_Initialize(blackHoleBinaryMergersMethod, Black_Hole_Binary_Merger_Do)
  implicit none
  type(varying_string), intent(in) :: blackHoleBinaryMergersMethod
  procedure(), pointer, intent(inout) :: Black_Hole_Binary_Merger_Do

  if (blackHoleBinaryMergersMethod == 'myMethod') Black_Hole_Binary_Merger_Do => My_Method_Do_Procedure
  return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the blackHoleBinaryMergersMethod input parameter. The procedure pointer Black_Hole_Binary_Merger_Do must be set to point to a function which returns the properties (mass and spin) of the merged black hole as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The cooling radius function must have the form:

subroutine My_Method_Do_Procedure(blackHoleMassA, blackHoleMassB, blackHoleSpinA, blackHoleSpinB, blackHoleMassFinal, blackHoleSpinFinal)
  implicit none
  double precision, intent(in) :: blackHoleMassA, blackHoleMassB, blackHoleSpinA, blackHoleSpinB
  double precision, intent(out) :: blackHoleMassFinal, blackHoleSpinFinal
  .
  .
  .
  return
end subroutine My_Method_Do_Procedure

The function must return the mass and spin (in blackHoleMassFinal and blackHoleSpinFinal respectively) of the black hole resulting from the merger of black holes with masses blackHoleMassA and blackHoleMassB and spins blackHoleSpinA and blackHoleSpinB. The subroutine should make no assumptions about the mass ordering of the input black holes (i.e. A could be more massive than B or vice versa).

Currently defined black hole binary merger methods are:

Rezzolla2008 Computes the properties of the merged black hole using the approximations of Rezzolla et al. [2008].

Chemical State

Additional methods for chemical states can be added using the chemicalStateMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the atomic_CIE_Cloudy method is described by a directive:

!# <chemicalStateMethod>
!# <unitName>Chemical_State_Atomic_CIE_Cloudy_Initialize</unitName>
!# </chemicalStateMethod>

Here, Chemical_State_Atomic_CIE_Cloudy_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

  implicit none
  type(varying_string), intent(in) :: chemicalStateMethod
  .
  .
  .
  return
end subroutine Method_Initialize
if (chemicalStateMethod == 'myMethod') then
    Electron_Density_Get => My_Method_Procedure
    Electron_Density_Temperature_Log_Slope_Get => My_Method_Temperature_Log_Slope_Procedure
    Chemical_Densities_Get => My_Method_Chemical_Densities_Procedure
end if
return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the chemicalStateMethod input parameter. The procedure pointer Electron_Density_Get must be set to point to a function which returns the electron density as described below. The other two electron density procedure pointers should point to functions which return the logarithmic gradients of the electron density with respect to temperature and density respectively. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.). The Chemical_Densities_Get pointer should be set to point to a subroutine that returns the densities of all “chemicals” active in the chemical subsystem (see §17.5.2).

The electron density function must have the form:

double precision function Electron_Density_Get(temperature,numberDensityHydrogen,abundances,radiation)
    implicit none
    double precision, intent(in) :: temperature,numberDensityHydrogen
    type(abundancesStructure), intent(in) :: abundances
    type(radiationStructure), intent(in) :: radiation

    return
end function Electron_Density_Get

The function must return the electron density (in units of cm$^{-3}$) for gas at the given temperature (in Kelvin), with hydrogen number density numberDensityHydrogen (in cm$^{-3}$), composition as described by the abundances structure and in the presence of a radiation field described by the radiation structure. The logarithmic slope functions should have the same template, but return the appropriate logarithmic slope instead.

The chemical densities subroutine must have the form:

subroutine Chemical_Densities_Get(theseAbundances,temperature,numberDensityHydrogen,abundances,radiation)
    implicit none
    type(chemicalAbundancesStructure), intent(inout) :: theseAbundances
    double precision, intent(in) :: temperature,numberDensityHydrogen
    type(abundancesStructure), intent(in) :: abundances
    type(radiationStructure), intent(in) :: radiation

    return
end subroutine Chemical_Densities_Get

The function must return the density (in units of cm$^{-3}$) of each chemical species for gas at the given temperature (in Kelvin), with hydrogen number density numberDensityHydrogen (in cm$^{-3}$), composi-
tion as described by the \texttt{abundances} structure and in the presence of a radiation field described by the \texttt{radiation} structure.

Currently defined chemical state methods are:

\texttt{CIE\_from\_file} Reads a tabulated CIE chemical state from a file and interpolates in the table to give a result. The XML file containing the table should have the following form:

\begin{verbatim}
<chemicalStates>
  <chemicalState>
    <temperature>
      <datum>10000.0</datum>
      <datum>15000.0</datum>
      ...
    </temperature>
    <electronDensity>
      <datum>1.0e-23</datum>
      <datum>1.7e-23</datum>
      ...
    </electronDensity>
    <hiDensity>
      <datum>0.966495864314214</datum>
      <datum>0.965828463162061</datum>
      ...
    </hiDensity>
    <hiiDensity>
      <datum>0.033504135685786</datum>
      <datum>0.0341715368379391</datum>
      ...
    </hiiDensity>
    <metallicity>-4.0</metallicity>
  </chemicalState>
  ...
</chemicalStates>
\end{verbatim}
Each chemicalState element should contain two lists (inside temperature and electronDensity tags) of datum elements which specify temperature (in Kelvin) and electron density (by number, relative to hydrogen) respectively, and a metallicity element which gives the logarithmic metallicity relative to Solar (a value of -999 or less is taken to imply zero metallicity). Optionally, hiDensity and hiiDensity elements may be added containing lists of H\textsubscript{i} and H\textsubscript{II} densities (by number, relative to hydrogen) respectively. Any number of coolingFunction elements may appear, but they must be in order of increasing metallicity and must all contain the same set of temperatures. The extrapolation element defines how the table is to be extrapolated in the low and high limits of temperature and metallicity. The method elements can take the following values:

**zero** The electron density is set to zero beyond the relevant limit.

**fixed** The electron density is held fixed at the value at the relevant limit.

**power law** The electron density is extrapolated assuming a power-law dependence beyond the relevant limit. This option is only allowed if the electron density is everywhere positive.

If the electron density is everywhere positive the interpolation will be done in the logarithmic of temperature, metallicity\(^4\) and electron density. Otherwise, interpolation is linear in these quantities. The electron density is scaled assuming a linear dependence on hydrogen density.

**atomicCIECloudy** Uses the Cloudy\(^5\) software to compute the chemical state for atomic gas in collisional ionization equilibrium. Cloudy will be downloaded, compiled and run automatically if necessary.

**PIE\textunderscore from\textunderscore file** Reads a tabulated PIE ionization state from a file and interpolates in the table to give a result. The HDF5 file containing the table should have the following form:

```plaintext
GROUP "/" {
    DATASET "neutralHydrogenRatio" {
        DATATYPE H5T_IEEE_F64BE
        DATASPACE SIMPLE { ( <ratioCount>, <redshiftCount>, <temperatureCount>, <densityCount> ) }
    }
}
```

\(^4\)The exception is if the first electron density is tabulated for zero metallicity. In that case, a linear interpolation in metallicity is always used between zero and the first non-zero tabulated metallicity.

\(^5\)Cloudy is used to generate a file which contains a tabulation of the chemical state suitable for reading by the CIE from file method. Generation of the tabulation typically takes several hours, but only needs to be done once as the stored table is simply read back in on later runs.
DATASET "density" {
    DATATYPE H5T_IEEE_F64BE
    DATASPACE SIMPLE { ( <densityCount> ) }
}

DATASET "electronRatio" {
    DATATYPE H5T_IEEE_F64BE
    DATASPACE SIMPLE { ( <ratioCount>, <redshiftCount>, <temperatureCount>, <densityCount> ) }
}

DATASET "heliumToHydrogenRatio" {
    DATATYPE H5T_IEEE_F64BE
    DATASPACE SIMPLE { ( <ratioCount> ) }
    ATTRIBUTE "extrapolationHigh" {
        DATATYPE H5T_STRING {}
        DATASPACE SCALAR
    }
    ATTRIBUTE "extrapolationLow" {
        DATATYPE H5T_STRING {}
        DATASPACE SCALAR
    }
}

DATASET "redshift" {
    DATATYPE H5T_IEEE_F64BE
    DATASPACE SIMPLE { ( <redshiftCount> ) }
    ATTRIBUTE "extrapolationHigh" {
        DATATYPE H5T_STRING {}
        DATASPACE SCALAR
    }
    ATTRIBUTE "extrapolationLow" {
        DATATYPE H5T_STRING {}
        DATASPACE SCALAR
    }
}

DATASET "temperature" {
    DATATYPE H5T_IEEE_F64BE
    DATASPACE SIMPLE { ( <temperatureCount> ) }
    ATTRIBUTE "extrapolationHigh" {
        DATATYPE H5T_STRING {}
        DATASPACE SCALAR
    }
    ATTRIBUTE "extrapolationLow" {
        DATATYPE H5T_STRING {}
        DATASPACE SCALAR
    }
}

The datasets should contain the following information:

**temperature** A list of temperatures (in units of Kelvin) at which cooling functions are tabulated.

If present, the *extrapolationLow* and *extrapolationHigh* attributes specify how the data should be extrapolated to lower and higher temperatures (see below);
redshift A list of redshifts at which cooling functions are tabulated. If present, the extrapolationLow and extrapolationHigh attributes specify how the data should be extrapolated to lower and higher redshifts (see below);

density A list of hydrogen number densities (in units of cm$^{-3}$) at which cooling functions are tabulated. If present, the extrapolationLow and extrapolationHigh attributes specify how the data should be extrapolated to lower and higher densities (see below);

eheliumToHydrogenRatio A list of helium-to-hydrogen number density ratios at which cooling functions are tabulated. If present, the extrapolationLow and extrapolationHigh attributes specify how the data should be extrapolated to lower and higher ratios (see below);

elements A list of the atomic numbers of elements for which cooling functions are tabulated;

neutralHydrogenRatio The neutral hydrogen fraction on the grid of temperature, density, redshift and helium-to-hydrogen number density ratio;

electronRatio The electron to hydrogen number density ratio on the grid of temperature, density, redshift and helium-to-hydrogen number density ratio.

Conditional Mass Functions

Additional methods for empirical conditional mass functions can be added using the conditionalMassFunctionMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Behroozi2010 method is described by a directive:

```plaintext
!# <conditionalMassFunctionMethod>
!# <unitName>Conditional_Mass_Functions_Behroozi2010_Initialize</unitName>
!# </conditionalMassFunctionMethod>
```

Here, Conditional_Mass_Functions_Behroozi2010_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
subroutine Method_Initialize(conditionalMassFunctionMethod&
   &, Cumulative_Conditional_Mass_Function_Get, Cumulative_Conditional_Mass_Function_Var_Get)
implicit none
   type(varying_string), intent(in) :: conditionalMassFunctionMethod
   procedure(double precision), pointer, intent(inout) :: Cumulative_Conditional_Mass_Function_Get, Cumulative_Conditional_Mass_Function_Var_Get

if (conditionalMassFunctionMethod == 'myMethod') then
   Cumulative_Conditional_Mass_Function_Get => My_Cumulative_Conditional_Mass_Function
   Cumulative_Conditional_Mass_Function_Var_Get => My_Cumulative_Conditional_Mass_Function_Var
   .
   .
end if
return
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the timePerTreeMethod input parameter. The procedure pointer Galacticus_Time_Per_Tree_Get must be set to point to a function which returns an estimate of the time taken (in seconds) to process a merger tree. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The functionw must have the form:
17. Adding New Methods

double precision function My_Cumulative_Conditional_Mass_Function(massHalo,mass)
  implicit none
  double precision, intent(in) :: massHalo,mass

  return
end function My_Cumulative_Conditional_Mass_Function_Var

double precision My_Cumulative_Conditional_Mass_Function_Var(massHalo,massLow,massHigh)
  implicit none
  double precision, intent(in) :: massHalo,massLow,massHigh

  return
end function My_Cumulative_Conditional_Mass_Function_Var

The first function must return the number of galaxies of mass greater than mass in halos of mass massHalo. The second function should return the variance in the number of galaxies in the mass range massLow to massHigh in halos of mass massHalo.

Currently defined tree timing methods are:

Behroozi2010 This method uses the fitting function of Behroozi et al. [2010] to compute the conditional mass function. To compute the variance in the mass function, this method assumes that the number of satellite galaxies follows a Poisson distribution, while central galaxies follow a Bernoulli distribution.

Cooling Rate

Additional methods for the cooling rate from the hot halo can be added using the coolingRateMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the White-Frenk1991 method is described by a directive:

!# <coolingRateMethod>
!# <unitName>Cooling_Rate_White_Frenk_Initialize</unitName>
!# </coolingRateMethod>

Here, Cooling_Rate_White_Frenk_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

subroutine Method_Initialize(coolingRateMethod,Cooling_Rate_Get)
  implicit none
  type(varying_string), intent(in) :: coolingRateMethod
  procedure(), pointer, intent(inout) :: Cooling_Rate_Get

  if (coolingRateMethod == 'myMethod') Cooling_Rate_Get => My_Method_Get
  return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the coolingRateMethod input parameter. The procedure pointer Cooling_Rate_Get must be set to point to a function which returns the
cooling rate from the hot halo. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The cooling rate function must have the form:

```fortran
double precision function Cooling_Rate_Get(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  .
  .
  return
end function Cooling_Rate_Get
```

The function must return the rate of mass drop-out from the hot halo (in units of $M/\text{Gyr}$) for `thisNode`.

Currently defined cooling rate methods are:

### White-Frenk1991
Implements something similar to that proposed by White and Frenk [1991]. Namely, the cooling rate is set equal to

$$
\dot{M}_{cool} = 4\pi \rho(r_{\text{infall}})^2 r_{\text{infall}}^2 (17.4)
$$

if the infall radius is within the outer radius of the hot halo and

$$
\dot{M}_{cool} = \frac{M_{\text{hot}}}{\tau_{\text{ dynamical, halo}}} (17.5)
$$

otherwise.

### Cole2000
Implements the cooling rate algorithm from Cole et al. [2000].

### simple
Implements a simple algorithm in which the cooling rate is determined from a fixed timescale.

### simpleScaling
Implements a simple algorithm in which the cooling rate is determined from a timescale which is a function of halo mass and redshift.

### Cooling Function

Additional methods for cooling functions can be added using the `coolingFunctionMethods`, `coolingFunctionCompute`, `coolingFunctionDensitySlopeCompute` and `coolingFunctionTemperatureSlopeCompute` directives. Each directive should contain a single argument, giving the name of a subroutine to be called to either initialize the method or compute the relevant quantity. For example, the `atomicCIECloudy` method is initialized by a directive:

```fortran
!# <coolingFunctionMethods>
!# <unitName>Cooling_Function_Atomic_CIE_Cloudy_Initialize</unitName>
!# </coolingFunctionMethods>
```

Here, `Cooling_Function_Atomic_CIE_Cloudy_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(coolingFunctionMethods,coolingFunctionsMatched)
  implicit none
  type(varying_string), intent(in ) :: coolingFunctionMethods(:)
  integer, intent(inout) :: coolingFunctionsMatched
  .
  return
end subroutine Method_Initialize
```
17. Adding New Methods

```fortran
if (any(coolingFunctionMethods == 'myMethod')) then
  .
  .
end if
return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `coolingFunctionMethods` input parameter. The initialization routine should record that whether this cooling function was selected, and perform any other initialization necessary. For each cooling function matched by this method, the value of `coolingFunctionsMatched` should be incremented by one—this permits a check that all cooling functions were matched.

The other directives should specify subroutines with the following template:

```fortran
subroutine Cooling_Function_PropertyCompute(coolingFunctionProperty, temperature, numberDensityHydrogen, &
  abundances, chemicalDensities, radiation)
implicit none
double precision, intent(in) :: temperature, numberDensityHydrogen
  type(abundancesStructure), intent(in) :: abundances
  type(chemicalAbundancesStructure), intent(in) :: chemicalDensities
  type(radiationStructure), intent(in) :: radiation
double precision, intent(out) :: coolingFunctionProperty
  .
  .
return
end subroutine Cooling_Function_PropertyCompute
```

and each should return the relevant quantity in `coolingFunctionProperty`. The `coolingFunctionCompute` subroutine should return the cooling function, the `coolingFunctionDensitySlopeCompute` subroutine should return the partial derivative with respect to hydrogen density and the `coolingFunctionTemperatureSlopeCompute` subroutine should return the partial derivative with respect to temperature.

Currently defined cooling function methods are:

- **CIE_from_file** Reads a tabulated CIE cooling function from a file and interpolates in the table to give a result. The XML file containing the table should have the following form:

```
<coolingFunctions>
  <coolingFunction>
    <temperature>
      <datum>10000.0</datum>
      <datum>15000.0</datum>
      .
      .
    </temperature>
    <coolingRate>
      <datum>1.0e-23</datum>
      <datum>1.7e-23</datum>
    </coolingRate>
  </coolingFunction>
</coolingFunctions>
```
Each coolingFunction element should contain two lists (inside temperature and coolingRate tags) of datum elements which specify temperature (in Kelvin) and cooling function (in ergs cm$^{-3}$ s$^{-1}$ computed for a hydrogen density of 1 cm$^{-3}$) respectively, and a metallicity element which gives the logarithmic metallicity relative to Solar (a value of -999 or less is taken to imply zero metallicity). Any number of coolingFunction elements may appear, but they must be in order of increasing metallicity and must all contain the same set of temperatures. The extrapolation element defines how the table is to be extrapolated in the low and high limits of temperature and metallicity. The method elements can take the following values:

- **zero** The cooling function is set to zero beyond the relevant limit.
- **fixed** The cooling function is held fixed at the value at the relevant limit.
- **power law** The cooling function is extrapolated assuming a power-law dependence beyond the relevant limit. This option is only allowed if the cooling function is everywhere positive.

If the cooling function is everywhere positive the interpolation will be done in the logarithmic of temperature, metallicity and cooling function. Otherwise, interpolation is linear in these quantities.

---

$^6$The exception is if the first cooling function is tabulated for zero metallicity. In that case, a linear interpolation in metallicity is always used between zero and the first non-zero tabulated metallicity.
17. Adding New Methods

Figure 17.1.: Cooling function for atomic gas in collisional ionization equilibrium computed using Cloudy 08.00.

The cooling function is scaled assuming a quadratic dependence on hydrogen density.

atomic_CIE_Cloudy  Uses the CLOUDY software to compute a cooling function for atomic gas in collisional ionization equilibrium. CLOUDY will be downloaded, compiled and run automatically if necessary. Figure 17.1 shows the cooling function from this method.

CMB_Compton  Computes the cooling function due to Compton scattering off of CMB photons.

molecularHydrogenGalliPalla  Implements the molecular hydrogen cooling function from Galli and Palla [1998].

Cloudy is used to generate a file which contains a tabulation of the cooling function suitable for reading by the CIE from file method. Generation of the tabulation typically takes several hours, but only needs to be done once as the stored table is simply read back in on later runs.
17.4. Existing Method Types

Cooling Radius

Additional methods for cooling radius calculations can be added using the `coolingRadiusMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `simple` method is described by a directive:

```plaintext
!# <coolingRadiusMethod>
!# <unitName>Cooling_Radius_Simple_Initialize</unitName>
!# </coolingRadiusMethod>
```

Here, `Cooling_Radius_Simple_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
subroutine Method_Initialize(coolingRadiusMethod,Cooling_Radius_Get,Cooling_Radius_Growth_Rate_Get)
implicit none
  type(varying_string), intent(in) :: coolingRadiusMethod
  procedure(), pointer, intent(inout) :: Cooling_Radius_Get,Cooling_Radius_Growth_Rate_Get

  if (coolingRadiusMethod == 'myMethod') then
    Cooling_Radius_Get => My_Method_Get_Procedure
    Cooling_Radius_Growth_Rate_Get => My_Method_Growth_Rate_Get_Procedure
  end if

end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `coolingRadiusMethod` input parameter. The procedure pointer `Cooling_Radius_Get` must be set to point to a function which returns the cooling function as described below while `Cooling_Radius_Growth_Rate_Get` should be set to point to a function which returns the rate at which the cooling radius is growing. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The cooling radius function must have the form:

```plaintext
double precision function Cooling_Radius_Get(thisNode)
implicit none
  type(treeNode), intent(inout), pointer :: thisNode

  return
end function Cooling_Radius_Get
```

The function must return the cooling radius (in units of Mpc) for `thisNode`. The cooling radius growth rate function should have the same template but return the rate at which the cooling radius grows in units of Mpc/Gyr.

Currently defined cooling radius methods are:

- **simple** Computes the cooling radius by seeking the radius at which the time available for cooling equals the cooling time. The growth rate is determined consistently based on the slope of the density profile, the density dependence of the cooling function and the rate at which the time available for cooling is increasing. This method assumes that the cooling time is a monotonic function of radius.

- **isothermal** Computes the cooling radius by assuming that the hot gas density profile is an isothermal profile \( \rho(r) \propto r^{-2} \), and that the cooling rate scales as density squared, \( \dot{E} \propto \rho^2 \), such that the cooling time scales as inverse density, \( t_{cool} \propto \rho^{-1} \). Consequently, the cooling radius grows as the square root of the time available for cooling.
17. Adding New Methods

**Cooling: Freefall Radius**

Additional methods for freefall radius in cooling calculations can be added using the `freefallRadiusMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `simple` method is described by a directive:

```
!# <freefallRadiusMethod>
!# <unitName>Freefall_Radius_Dark_Matter_Halo_Initialize</unitName>
!# </freefallRadiusMethod>
```

Here, `Freefall_Radius_Dark_Matter_Halo_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(freefallRadiusMethod,Freefall_Radius_Get,Freefall_Radius_Growth_Rate_Get)
implicit none
    type(varying_string), intent(in) :: freefallRadiusMethod
    procedure(), pointer, intent(inout) :: Freefall_Radius_Get, Freefall_Radius_Growth_Rate_Get

    if (freefallRadiusMethod == 'myMethod') then
        Freefall_Radius_Get => My_Method_Get_Procedure
        Freefall_Radius_Growth_Rate_Get => My_Method_Growth_Rate_Get_Procedure
    end if
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `freefallRadiusMethod` input parameter. The procedure pointer `Freefall_Radius_Get` must be set to point to a function which returns the freefall radius as described below while `Freefall_Radius_Growth_Rate_Get` should be set to point to a function which returns the rate at which the freefall radius is growing. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The freefall radius function must have the form:

```fortran
double precision function Freefall_Radius_Get(thisNode)
    implicit none
    type(treeNode), intent(inout), pointer :: thisNode

    return
end function Freefall_Radius_Get
```

The function must return the freefall radius (in units of Mpc) for `thisNode`. The freefall radius growth rate function should have the same template but return the rate at which the freefall radius grows in units of Mpc/Gyr.

Currently defined freefall radius methods are:

- **darkMatterHalo** Computes the freefall radius by finding the radius in the dark matter halo profile from which a test particle could have free-fallen to zero radius (assuming it began at rest) in the time available for freefall.

**Cooling: Infall Radius**

Additional methods for the infall radius in cooling calculations can be added using the `infallRadiusMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `coolingRadius` method is described by a directive:
17.4. Existing Method Types

Here, \texttt{Infall\_Radius\_Cooling\_Radius\_Initialize} is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

\begin{verbatim}
subroutine Method\_Initialize(infallRadiusMethod,Infall\_Radius\_Get,Infall\_Radius\_Growth\_Rate\_Get)
implicit none
  type(varying\_string), intent(in) :: infallRadiusMethod
  procedure(), pointer, intent(inout) :: Infall\_Radius\_Get, Infall\_Radius\_Growth\_Rate\_Get

  if (infallRadiusMethod == 'myMethod') then
    Infall\_Radius\_Get => My\_Method\_Get\_Procedure
    Infall\_Radius\_Growth\_Rate\_Get => My\_Method\_Growth\_Rate\_Get\_Procedure
  end if

  return
end subroutine Method\_Initialize
\end{verbatim}

where \texttt{myMethod} is the name of this method as will be specified by the \texttt{infallRadiusMethod} input parameter. The procedure pointer \texttt{Infall\_Radius\_Get} must be set to point to a function which returns the infall radius as described below while \texttt{Infall\_Radius\_Growth\_Rate\_Get} should be set to point to a function which returns the rate at which the infall radius is growing. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The infall radius function must have the form:

\begin{verbatim}
double precision function Infall\_Radius\_Get(thisNode)
implicit none
  type(treeNode), intent(inout), pointer :: thisNode

  return
end function Infall\_Radius\_Get
\end{verbatim}

The function must return the infall radius (in units of Mpc) for \texttt{thisNode}, i.e. the radius from which gas in the hot halo that is currently accreting onto the galaxy originated. The infall radius growth rate function should have the same template but return the rate at which the infall radius grows in units of \text{Mpc/Gyr}.

Currently defined infall radius methods are:

\begin{itemize}
  \item \texttt{coolingRadius} Assumes that the infall radius equals the cooling radius.
  \item \texttt{cooling and freefall} Assumes that the infall radius is equal to the smaller of the cooling and freefall radii.
\end{itemize}

\textbf{Cooling Specific Angular Momentum}

Additional methods for calculations of the specific angular momentum of cooling gas can be added using the \texttt{coolingSpecificAngularMomentumMethod} directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the \texttt{simple} method is described by a directive:
17. Adding New Methods

Here, Cooling_Time_Simple.Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(coolingSpecificAngularMomentumMethod, Cooling_Specific_Angular_Momentum_Get)
  implicit none
  type(varying_string), intent(in) :: coolingSpecificAngularMomentumMethod
  procedure(), pointer, intent(inout) :: Cooling_Specific_Angular_Momentum_Get

  if (coolingSpecificAngularMomentumMethod == 'myMethod') then
    Cooling_Specific_Angular_Momentum_Get => My_Method_Get_Procedure
  end if
  return
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the coolingSpecificAngularMomentumMethod input parameter. The procedure pointer Cooling_Specific_Angular_Momentum_Get must be set to point to a function which returns the specific angular momentum of cooling gas. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The specific angular momentum of cooling gas function must have the form:

```fortran
double precision function Cooling_Specific_Angular_Momentum_Get(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode

  return
end function Cooling_Specific_Angular_Momentum_Get
```

The function must return the specific angular momentum (in units of km/s Mpc) of gas that is cooling in thisNode.

Currently defined specific angular momentum of cooling gas methods are:

- **constantRotation** Computes the specific angular momentum of the cooling gas based on the cooling radius, mean specific angular momentum and the assumption of a constant mean rotation speed in the cooling gas as a function of radius.

- **mean** Assumes that the specific angular momentum of the cooling gas always equals the mean specific angular momentum of the hot halo.

**Cooling Time Available**

Additional methods for the time available for cooling can be added using the coolingTimeAvailableMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the White-Frenk method is described by a directive:

```fortran
!# <coolingTimeAvailableMethod>
!# <unitName>Cooling_Time_Available_WF_Initialize</unitName>
!# </coolingTimeAvailableMethod>
```
17.4. Existing Method Types

Here, Cool_Time_Available_WF.Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(coolingTimeAvailableMethod, Cooling_Time_Available_Get&, Cooling_Time_Available_Increase_Rate_Get)
  implicit none
  type(varying_string), intent(in) :: coolingTimeAvailableMethod
  procedure(), pointer, intent(inout) :: Cooling_Time_Available_Get, Cooling_Time_Available_Increase_Rate_Get

  if (coolingTimeAvailableMethod == 'myMethod') then
    Cooling_Time_Available_Get => My_Method_Get
    Cooling_Time_Available_Increase_Rate_Get => My_Method_Increase_Rate_Get
  end if
  return
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the coolingTimeAvailableMethod input parameter. The procedure pointers Cooling_Time_Available_Get and Cooling_Time_Available_Increase_Rate_Get must be set to point to functions which return the time available for cooling and the rate of increase of this time respectively. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The cooling time available functions must have the form:

```fortran
double precision function Cooling_Time_Available_Get(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  .
  .
  .
  return
end function Cooling_Time_Available_Get
```

The first function must return the time available for cooling (in units of Gyr) for thisNode, while the second must return the rate of increase of this time.

Currently defined cooling time available methods are:

- **White-Frenk** The time available is set to a value between the age of the Universe and the dynamical time of the halo, depending on the interpolating parameter [coolingTimeAvailableAgeFactor];
- **haloFormation** The time available for cooling is set equal to the current time minus the formation time of the halo.

**Cooling Time Available For Freefall**

Additional methods for the time available for freefall in cooling calculations can be added using the freeFallTimeAvailableMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the haloFormation method is described by a directive:

```plaintext
!# <freeFallTimeAvailableMethod>
!# <unitName>FreeFall_Time_Available_Halo_Formation.Initialize</unitName>
!# </freeFallTimeAvailableMethod>
```
Here, `Freefall_Time_Available_Halo_Formation.Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(freefallTimeAvailableMethod,Freefall_Time_Available_Get&,Freefall_Time_Available_Increase_Rate_Get)
    implicit none
    type(varying_string), intent(in) :: freefallTimeAvailableMethod
    procedure(), pointer, intent(inout) :: Freefall_Time_Available_Get,Freefall_Time_Available_Increase_Rate_Get
    if (freefallTimeAvailableMethod == 'myMethod') then
        Freefall_Time_Available_Get => My_Method_Get
        Freefall_Time_Available_Increase_Rate_Get => My_Method_Increase_Rate_Get
    end if
    return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `freefallTimeAvailableMethod` input parameter. The procedure pointers `Freefall_Time_Available_Get` and `Freefall_Time_Available_Increase_Rate_Get` must be set to point to functions which return the time available for freefall in cooling calculations and the rate of increase of this time respectively. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The freefall time available functions must have the form:

```fortran
double precision function Freefall_Time_Available_Get(thisNode)
    implicit none
    type(treeNode), intent(inout), pointer :: thisNode
    
    return
end function Freefall_Time_Available_Get
```

The first function must return the time available for freefall cooling calculations (in units of Gyr) for `thisNode`, while the second must return the rate of increase of this time.

Currently defined freefall time available methods are:

```markdown
**haloFormation**  The time available for cooling is set equal to the current time minus the formation time of the halo.
```

**Cooling Time**

Additional methods for cooling time calculations can be added using the `coolingTimeMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `simple` method is described by a directive:

```markdown
!# <coolingTimeMethod>
!# <unitName>Cooling_Time_Simple_Initialize</unitName>
!# </coolingTimeMethod>
```

Here, `Cooling_Time_Simple_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:
subroutine Method_Initialize(coolingTimeMethod,Cooling_Time_Get,Cooling_Time_Density_Log_Slope_Get,Cooling_Time_Temperature_Log_Slope_Get)
  implicit none
  type(varying_string), intent(in) :: coolingTimeMethod
  procedure(), pointer, intent(inout) :: Cooling_Time_Get,Cooling_Time_Density_Log_Slope_Get,Cooling_Time_Temperature_Log_Slope_Get

  if (coolingTimeMethod == 'myMethod') then
    Cooling_Time_Get => My_Method_Get_Procedure
    Cooling_Time_Density_Log_Slope_Get => My_Method_Density_Log_Slope_Procedure
    Cooling_Time_Temperature_Log_Slope_Get => My_Method_Temperature_Log_Slope_Procedure
  end if
  return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the coolingTimeMethod input parameter. The procedure pointer Cooling_Time_Get must be set to point to a function which returns the cooling function as described below while the other two pointers should point to functions which return the appropriate logarithmic slope. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The cooling time function must have the form:

double precision function Cooling_Time_Get(temperature,density,abundances,chemicalDensities,radiation)
  implicit none
  double precision, intent(in) :: temperature,density
  type(abundancesStructure), intent(in) :: abundances
  type(chemicalAbundancesStructure), intent(in) :: chemicalDensities
  type(radiationStructure), intent(in) :: radiation

  return
end function Cooling_Time_Get

The function must return the cooling time (in units of Gyr) for at the specified temperature, density and for composition and radiation field as specified by the abundances, chemicalDensities and radiation structures. The logarithmic slope functions should have the same template, but return the logarithmic slope of the cooling time with respect to the appropriate variable instead.

Currently defined cooling time methods are:

**simple** Compute the cooling time as the ratio of the gas thermal energy density to the volume rate of radiative energy loss. The gas is assumed to have an effective number of degrees of freedom specified by the coolingTimeSimpleDegreesOfFreedom parameter.

**Cosmological Mass Root Variance**

Additional methods for computing the cosmological mass root variance, \( \sigma(M) \), can be added using the cosmologicalMassVarianceMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the filteredPowerSpectrum method is described by a directive:

```plaintext
!# <cosmologicalMassVarianceMethod>
!# <unitName>Cosmological_Mass_Variance_Filtered_Power_SpectrumInitializer</unitName>
!# </cosmologicalMassVarianceMethod>
```
17. Adding New Methods

Here, Cosmological_Mass_Variance_Filtered_Power_Spectrum.Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(cosmologicalMassVarianceMethod, Cosmological_Mass_Variance_Tabulate)
  implicit none
  type (varying_string), intent(in) :: cosmologicalMassVarianceMethod
  procedure( , pointer, intent(inout) :: Cosmological_Mass_Variance_Tabulate

  if (cosmologicalMassVarianceMethod == 'myMethod') then
    Cosmological_Mass_Variance_Tabulate => My_Method_Tabulate
  .

  end if
  return
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the cosmologicalMassVarianceMethod input parameter. The procedure pointer Cosmological_Mass_Variance_Tabulate must be set to point to a function which populates a table1D object with a tabulation of $\sigma(M)$. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The tabulation function must have the form:

```fortran
subroutine Cosmological_Mass_Variance_Filtered_Power_Spectrum(mass, massNormalization, sigmaNormalization, sigmaTable)
  implicit none
  double precision , intent(in) :: mass, massNormalization
  double precision , intent(inout) :: sigmaNormalization
  class (table1D), intent(inout), allocatable :: sigmaTable

  return
end subroutine Cosmological_Mass_Variance_Filtered_Power_Spectrum
```

The function should allocate sigmaTable to a suitable type of table1D object and populate it with a tabulation of $\sigma(M)$ which includes the given mass. On input, the required normalization of $\sigma(M)$ at mass massNormalization is given by sigmaNormalization. The function should divide this value by the unnormalized value of $\sigma(M)$—this is used to normalize the cosmological power spectrum.

Currently defined cosmological mass root variance methods are:

- filteredPowerSpectrum The mass root variance is found by integrating over the transferred linear power spectrum multiplied by the selected window function (see §17.4.1).

Critical Overdensity for Halo Collapse

Additional methods for the critical linear theory overdensity for halo collapse can be added using the criticalOverdensityMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the sphericalTopHat method is described by a directive:

```fortran
! # <criticalOverdensityMethod>
! # <unitName>Spherical_Collape_Delta_Critical_Initialize</unitName>
! # </criticalOverdensityMethod>
```
Here, Spherical_Collapse_Delta_CriticalInitialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(criticalOverdensityMethod,Critical_Overdensity_Tabulate)
  implicit none
  type(varying_string), intent(in) :: criticalOverdensityMethod
  procedure(), pointer, intent(inout) :: Critical_Overdensity_Tabulate
  if (criticalOverdensityMethod.eq.'myMethod') then
    Critical_Overdensity_Tabulate => My_Do_Tabulate
  else
  end if
  return
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the criticalOverdensityMethod input parameter. The procedure pointer Critical_Overdensity_Tabulate must be set to point to a subroutine which tabulates the critical overdensity as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.). The tabulation subroutine must have the form:

```fortran
subroutine Critical_Overdensity_Tabulate(time,deltaCritNumberPoints,deltaCritTime,deltaCritDeltaVirial)
  implicit none
  double precision, intent(in) :: time
  integer, intent(out) :: deltaCritNumberPoints
  double precision, intent(inout), allocatable, dimension(:) :: deltaCritTime,deltaCritDeltaVirial
  return
end subroutine Critical_Overdensity_Tabulate
```

The subroutine must tabulate the critical overdensity in array deltaCritDeltaVirial() as a function of wavenumber deltaCritTime() (these arrays must be allocated to the correct size, and may be previously allocated, therefore requiring a deallocation). The number of tabulated points should be returned in deltaCritNumberPoints. The subroutine should ensure that the currently requested time is within the range of the tabulated function (preferably with some buffer).

Currently defined critical overdensity methods are:

- **sphericalTopHat**  The critical overdensity is computed for a Universe containing collisionless matter and a cosmological constant following the spherical top hat collapse model (see, for example, Percival 2005).

- **Kitayama-Suto1996**  The critical overdensity is computed using the fitting formula of Kitayama and Suto [1996], and is therefore valid only for flat cosmological models.

**Critical Overdensity for Halo Collapse: Mass Scaling**

Additional methods for the mass scaling of the critical linear theory overdensity for halo collapse can be added using the criticalOverdensityMassScalingMethod directive. The directive should contain a
17. Adding New Methods

single argument, giving the name of a subroutine to be called to initialize the method. For example, the warm dark matter method is described by a directive:

```plaintext
!# <criticalOverdensityMassScalingMethod>
!# <unitName>Critical_Overdensity_Mass_Scaling_WDM_Initialize</unitName>
!# </criticalOverdensityMassScalingMethod>
```

Here, `Critical_Overdensity_Mass_Scaling_WDM_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
subroutine Method_Initialize(criticalOverdensityMassScalingMethod, &
implicit none
  type(varying_string), intent(in) :: criticalOverdensityMassScalingMethod
  procedure(), pointer, intent(inout) :: Critical_Overdensity_Mass_Scaling_Get, &
& Critical_Overdensity_Mass_Scaling_Gradient_Get
if (criticalOverdensityMassScalingMethod == 'myMethod') then
    Critical_Overdensity_Mass_Scaling_Get => My_Do_Tabulate
    Critical_Overdensity_Mass_Scaling_Gradient_Get => My_Do_Gradient_Tabulate
    ...
end if
return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `criticalOverdensityMassScalingMethod` input parameter. The procedure pointers `Critical_Overdensity_Mass_Scaling_Get` and `Critical_Overdensity_Mass_Scaling_Gradient_Get` must be set to point to functions which return the critical overdensity mass scaling and its gradient as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The mass scaling function must have the form:

```plaintext
double precision function Critical_Overdensity_Mass_Scaling_Get(mass)
implicit none
  double precision, intent(in) :: mass
  ...
return
end function Critical_Overdensity_Mass_Scaling_Get
```

The function should return the factor by which the critical overdensity for collapse at the given mass scale (given in units of $M_\odot$) differs from that for the case $M \to \infty$. The mass scaling gradient function should have the same form, but should return the derivative of the scaling with respect to mass.

Currently defined critical overdensity mass scaling methods are:

- **null** The critical overdensity is assumed to have no scaling with mass;
- **warm dark matter** The mass scaling is computed for warm dark matter using a fitting function to the results of Barkana et al. [2001].
**Dark Matter Density Profile**

Additional methods for the dark matter density profile can be added using the `darkMatterProfileMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the NFW method is described by a directive:

```
!# <darkMatterProfileMethod>
!# <unitName>Dark_Matter_Profile_NFW_Initialize</unitName>
!# </darkMatterProfileMethod>
```

Here, `Dark_Matter_Profile_NFW_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(darkMatterProfileMethod,Dark_Matter_Profile_Density_Get,Dark_Matter_Profile_Energy_Get,&
& Dark_Matter_Profile_Energy_Growth_Rate_Get,Dark_Matter_Profile_Rotation_Normalization_Get,&
& Dark_Matter_Profile_Radius_from_Specific_Angular_Momentum_Get,Dark_Matter_Profile_Circular_Velocity_Get,&
& Dark_Matter_Profile_Freefall_Radius_Get,Dark_Matter_Profile_Freefall_Radius_Increase_Rate_Get)
implicit none

type(varying_string), intent(in) :: darkMatterProfileMethod
procedure(), pointer, intent(inout) :: Dark_Matter_Profile_Density_Get, Dark_Matter_Profile_Energy_Get,&
& Dark_Matter_Profile_Energy_Growth_Rate_Get, Dark_Matter_Profile_Rotation_Normalization_Get,&
& Dark_Matter_Profile_Radius_from_Specific_Angular_Momentum_Get, Dark_Matter_Profile_Circular_Velocity_Get,&
& Dark_Matter_Profile_Freefall_Radius_Get, Dark_Matter_Profile_Freefall_Radius_Increase_Rate_Get

if (darkMatterProfileMethod == 'myMethod') then
  Dark_Matter_Profile_Density_Get => My_Dark_Matter_Profile_Density
  Dark_Matter_Profile_Energy_Get => My_Dark_Matter_Profile_Energy
  Dark_Matter_Profile_Energy_Growth_Rate_Get => My_Dark_Matter_Profile_Energy_Growth_Rate
  Dark_Matter_Profile_Rotation_Normalization_Get => My_Dark_Matter_Profile_Rotation_Normalization
  Dark_Matter_Profile_Radius_from_Specific_Angular_Momentum_Get => My_Radius_from_Specific_Angular_Momentum
  Dark_Matter_Profile_Circular_Velocity_Get => My_Dark_Matter_Profile_Circular_Velocity
  Dark_Matter_Profile_Potential_Get => My_Dark_Matter_Profile_Potential
  Dark_Matter_Profile_Enclosed_Mass_Get => My_Dark_Matter_Profile_Enclosed_Mass
  Dark_Matter_Profile_kSpace_Get => My_Dark_Matter_Profile_kSpace
  Dark_Matter_Profile_Freefall_Radius_Get => My_Dark_Matter_Profile_Freefall_Radius
  Dark_Matter_Profile_Freefall_Radius_Increase_Rate_Get => My_Dark_Matter_Profile_Freefall_Radius_Increase_Rate
end if
return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `darkMatterProfileMethod` input parameter. The procedure pointers must be set to point to functions which return properties of the dark matter density profile as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The density, enclosed mass, potential and circular velocity functions must have the form:

```fortran
double precision function My_Dark_Matter_Profile_Property(thisNode,radius)
```

781
17. Adding New Methods

```fortran
implicit none
type(treeNode), intent(inout), pointer :: thisNode
double precision, intent(in) :: radius

return
end function My_Dark_Matter_Profile_Property
```

These functions should compute and return the density (in units of \(M_\odot\) Mpc\(^{-3}\)), enclosed dark matter mass (in units of \(M_\odot\)), gravitational potential (in units of \((\text{km/s})^2\)) and circular velocity (in units of km/s) due to dark matter at the given radius (in units of Mpc) for thisNode respectively.

The freefall radius functions must have the form:

```fortran
double precision function My_Dark_Matter_Profile_Property(thisNode, time)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
double precision, intent(in) :: time

  return
end function My_Dark_Matter_Profile_Property
```

These functions should compute and return the freefall radius (in units of Mpc), or its growth rate given a time available for freefall (in Gyr) for thisNode respectively.

The energy, energy growth rate and rotation velocity normalization functions must have the form:

```fortran
double precision function My_Dark_Matter_Profile_Property(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode

  return
end function My_Dark_Matter_Profile_Property
```

The energy functions should compute and return the energy (potential plus kinetic; in units of \(M_\odot\) (km/s)^2) of the dark matter halo out to the virial radius of thisNode and the rate of change of that energy (in units of \(M_\odot\) (km/s)^2 Gyr\(^{-1}\)) respectively. The rotation normalization function should compute and return the normalization between rotation speed and mean specific angular momentum (in units of Mpc\(^{-1}\)) of thisNode assuming that the dark matter halo rotates at the same velocity at all radii.

Finally, the radius from specific angular momentum function must have the form:

```fortran
double precision function My_Dark_Matter_Profile_Radius_From_Specific_Angular_Momentum(thisNode, specificAngularMomentum)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
double precision, intent(in) :: specificAngularMomentum

  return
end function My_Dark_Matter_Profile_Radius_From_Specific_Angular_Momentum
```
This function should compute and return the radius (in units of Mpc) in thisNode at which a circular orbit would have the given specificAngularMomentum (in units of km/s Mpc).

The “kSpace” function must have the form:

```fortran
double precision function My_Dark_Matter_Profile_kSpace(thisNode,wavenumber)
    implicit none
    type(treeNode), intent(inout), pointer :: thisNode
    double precision, intent(in) :: wavenumber

    return
end function My_Dark_Matter_Profile_kSpace
```

This function should compute and return the Fourier transform of the dark matter halo density profile (normalized to unity at small wavenumber—as defined in Cooray and Sheth 2002) for thisNode at the given wavenumber (specified in Mpc$^{-1}$).

Currently defined dark matter density profile methods are:

- **Isothermal** The density profile is a singular isothermal sphere;
- **NFW** The density profile proposed by Navarro et al. [1997].
- **Einasto** The Einasto density profile, described, for example, by Cardone et al. [2005].

### Dark Matter Halo Mass Accretion History

Additional methods for dark matter halo mass accretion histories can be added using the `darkMatterAccretionHistoryMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Zhao2009 method is described by a directive:

```plaintext
!# <darkMatterAccretionHistoryMethod>
!# <unitName>Dark_Matter_Mass_Accretion_Zhao2009_Initialize</unitName>
!# </darkMatterAccretionHistoryMethod>
```

Here, `Dark_Matter_Mass_Accretion_Zhao2009_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(darkMatterAccretionHistoryMethod,Dark_Matter_Halo_Mass_Accretion_Time_Get)
    implicit none
    type(varying_string), intent(in) :: darkMatterAccretionHistoryMethod
    procedure(), pointer, intent(inout) :: Dark_Matter_Halo_Mass_Accretion_Time_Get

    if (darkMatterAccretionHistoryMethod == 'myMethod') then
        Dark_Matter_Halo_Mass_Accretion_Time_Get => My_Time_Get
    end if
    return
end subroutine Method_Initialize
```
where myMethod is the name of this method as will be specified by the darkMatterAccretionHistoryMethod input parameter. The procedure pointer Dark_Matter_Halo_Mass_Accretion_Time_Get must be set to point to a function which returns the time at which a given mass is reached in the mass accretion history. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The time function must have the form:

```fortran
double precision function My_Dark_Matter_Halo_Mass_Accretion_Time(baseNode,nodeMass)  
    implicit none  
    type(treeNode), intent(inout), pointer :: baseNode  
    double precision, intent(in) :: nodeMass  
    .  
    .  
    return  
end function My_Dark_Matter_Halo_Mass_Accretion_Time
```

The function should compute and return the time at which the mass accretion history of baseNode reaches the specified nodeMass.

Currently defined mass accretion history methods are:

- **Wechsler2002**: Uses the fitting function from Wechsler et al. [2002] to compute the mass accretion history. If [accretionHistoryWechslerFormationRedshiftCompute] is set to true then the formation redshift for each history is set using the method of Bullock et al. [2001], otherwise it can be set directly via the [accretionHistoryWechslerFormationRedshift] parameter;

- **Zhao2009**: Uses the algorithm of Zhao et al. [2009] to compute the mass accretion history.

**Dark Matter Density Profile Shape**

Additional methods for the dark matter density profile shape parameter can be added using the darkMatterShapeMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Gao2008 method is described by a directive:

```fortran
!# <darkMatterShapeMethod>
!# <unitName>Dark_Matter_Shapes_Gao20008_Initialize</unitName>
!# </darkMatterShapeMethod>
```

Here, Dark_Matter_Shapes_Gao20008_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(darkMatterShapeMethod,Dark_Matter_Profile_Shape_Get)  
    implicit none  
    type(varying_string), intent(in) :: darkMatterShapeMethod  
    procedure(), pointer, intent(inout) :: Dark_Matter_Profile_Shape_Get  
    if (darkMatterShapeMethod == 'myMethod') then  
        Dark_Matter_Profile_Shape_Get => My_Shape_Get  
    .  
    .  
    end if  
    return  
end subroutine Method_Initialize
```
where \texttt{myMethod} is the name of this method as will be specified by the \texttt{darkMatterShapeMethod} input parameter. The procedure pointer \texttt{Dark\_Matter\_Profile\_Shape\_Get} must be set to point to a function which returns the shape parameter of a node. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The shape parameter function must have the form:

\begin{verbatim}
double precision function My_Dark_Matter_Profile_Shape(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  .
  .
  .
  return
end function My_Dark_Matter_Profile_Shape
\end{verbatim}

The function should compute and return the shape parameter for \texttt{thisNode}.

Currently defined dark matter profile shape parameter methods are:

\textbf{Gao2008} The shape parameter is computed using a fitting function from Gao et al. [2008] - see §13.8.4 for details.

\textbf{Dark Matter Halo Spin Distribution}

Additional methods for the dark matter density profile concentration can be added using the \texttt{haloSpinDistributionMethod} directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the \texttt{Gao2008} method is described by a directive:

\begin{verbatim}
!# <haloSpinDistributionMethod>
!# <unitName>Halo_Spin_Distribution_Bett2007_Initialize</unitName>
!# </haloSpinDistributionMethod>
\end{verbatim}

Here, \texttt{Halo\_Spin\_Distribution\_Bett2007\_Initialize} is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

\begin{verbatim}
subroutine Method_Initialize(haloSpinDistributionMethod,Halo_Spin_Sample_Get)
  implicit none
  type(varying_string), intent(in) :: haloSpinDistributionMethod
  procedure(), pointer, intent(inout) :: Halo_Spin_Sample_Get

  if (haloSpinDistributionMethod == 'myMethod') then
    Halo_Spin_Sample_Get => My_Spin_Sample_Get
  .
  .
  .
  end if
  return
end subroutine Method_Initialize
\end{verbatim}

where \texttt{myMethod} is the name of this method as will be specified by the \texttt{haloSpinDistributionMethod} input parameter. The procedure pointer \texttt{Halo\_Spin\_Sample\_Get} must be set to point to a function which returns a spin parameter drawn at random from a distribution. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The spin parameter function must have the form:

\begin{verbatim}
subroutine My_Spin_Sample_Get()
  implicit none
  .
  .
  .
  return
end subroutine My_Spin_Sample_Get
\end{verbatim}
17. Adding New Methods

```plaintext
double precision function My_Spin_Distribution_Sample(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  .
  .
  return
end function My_Spin_Distribution_Sample
```

The function should compute and return a spin parameter for `thisNode` drawn at random from a distribution.

Currently defined spin distribution methods are:

- **lognormal** The spin is drawn from a lognormal distribution with median `[lognormalSpinDistributionMedian]` and width `[lognormalSpinDistributionSigma]`.
- **Bett2007** The spin is drawn from the distribution found by Bett et al. [2007]. The $\lambda_0$ and $\alpha$ parameter of Bett et al.’s distribution are set by the `[spinDistributionBett2007Lambda0]` and `[spinDistributionBett2007Alpha]` input parameters.
- **deltaFunction** The spin is drawn from a delta function distribution, i.e. a value equal to `[deltaFunctionSpinDistributionSpin]` is always returned.

## Dark Matter Halo Mass Loss Rates

Additional methods for dark matter halo mass loss rates can be added using the `darkMatterHaloMassLossRateMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `van den Bosch 2005` method is described by a directive:

```plaintext
!# <darkMatterHaloMassLossRateMethod>
!#  <unitName>Dark_Matter_Halos_Mass_Loss_Rate_vanDenBosch_Initialize</unitName>
!# </darkMatterHaloMassLossRateMethod>
```

Here, `Dark_Matter_Halos_Mass_Loss_Rate_vanDenBosch_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
subroutine Method_Initialize(darkMatterHaloMassLossRateMethod,Dark_Matter_Halos_Mass_Loss_Rate_Get)
  implicit none
  type(varying_string), intent(in) :: darkMatterHaloMassLossRateMethod
  procedure(), pointer, intent(inout) :: Dark_Matter_Halos_Mass_Loss_Rate_Get
  if (darkMatterHaloMassLossRateMethod == 'myMethod') then
    Dark_Matter_Halos_Mass_Loss_Rate_Get => My_Mass_Loss_Rate_Get
  .
  .
end if
return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `darkMatterHaloMassLossRateMethod` input parameter. The procedure pointer `Dark_Matter_Halos_Mass_Loss_Rate_Get` must be set to point to a function which returns a spin parameter drawn at random from a distribution. The initialization
subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The mass loss rate function must have the form:

```fortran
double precision function My_Mass_Loss_Rate(thisNode)
    implicit none
    type(treeNode), intent(inout), pointer :: thisNode
    .
    .
    return
end function My_Mass_Loss_Rate
```

The function should compute and return the mass loss rate from `thisNode` in units of $M_\odot$/Gyr.

Currently defined halo mass loss rate methods are:

- **null** Always returns zero mass loss rate.
- **vanDenBosch2005** Uses the algorithm of van den Bosch et al. [2005] to compute the mass loss rate.

### Excursion Set Barrier

Additional methods for the excursion set barrier can be added using the `excursionSetBarrierMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `linear` method is described by a directive:

```fortran
!# <excursionSetBarrierMethod>
!# <unitName>Excursion_Sets_Barriers_Linear_Initialize</unitName>
!# </excursionSetBarrierMethod>
```

Here, `Excursion_Sets_Barriers_Linear_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(excursionSetBarrierMethodExcursion_Sets_Barrier_Get,Excursion_Sets_Barrier_Gradient_Get,barrierName)
    implicit none
    type (varying_string ), intent(in ) :: excursionSetBarrierMethod
    procedure(double precision),pointer, intent(inout) :: Excursion_Sets_Barrier_Get,Excursion_Sets_Barrier_Gradient_Get
    type(varying_string), intent(inout) :: barrierName
    if (excursionSetBarrierMethod == 'myMethod') then
        Excursion_Sets_Barrier_Get => My_Barrier
        Excursion_Sets_Barrier_Gradient_Get => My_Barrier_Gradient
        barrierName=barrierName//":myLabel"
    end if
    return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `excursionSetBarrierMethod` input parameter. The procedure pointers `Excursion_Sets_Barrier_Get` and `Excursion_Sets_Barrier_Gradient_Get` must be set to point functions which return the barrier and its gradient respectively, as described below. The initialization subroutine should also append a descriptive label to the `barrierName`.
17. Adding New Methods

The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The barrier and barrier gradient functions must have the form:

double precision function Excursion_Sets_Barrier(variance, time)
  implicit none
  double precision, intent(in) :: variance, time
  .
  .
  return
end function Excursion_Sets_Barrier

The barrier function must return the barrier at the specified variance and time, while the barrier gradient function should return the derivative with respect to variance of the same barrier.

Currently defined excursion set barrier methods are:

- **linear** A linear (1st-order polynomial) barrier;
- **quadratic** A quadratic (2nd-order polynomial);
- **criticalOverdensity** A barrier equal to the critical overdensity for halo collapse.

**Excursion Set Barrier First Crossing Distribution**

Additional methods for the excursion set barrier first crossing distribution can be added using the `excursionSetFirstCrossingMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `linearBarrier` method is described by a directive:

```fortran
!# <excursionSetFirstCrossingMethod>
!# <unitName>Excursion_Sets_First_Crossing_Linear_Barrier_Initialize</unitName>
!# </excursionSetFirstCrossingMethod>
```

Here, `Excursion_Sets_First_Crossing_Linear_Barrier_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(excursionSetFirstCrossingMethod, Excursion_Sets_First_Crossing_Probability_Get, & Excursion_Sets_First_Crossing_Rate_Get, Excursion_Sets_Non_Crossing_Rate_Get)
  implicit none
  type (varying_string), intent(in) :: excursionSetFirstCrossingMethod
  procedure(double precision), pointer, intent(inout) :: Excursion_Sets_First_Crossing_Probability_Get, &
  Excursion_Sets_First_Crossing_Rate_Get, Excursion_Sets_Non_Crossing_Rate_Get
  if (excursionSetFirstCrossingMethod == 'myMethod') then
    Excursion_Sets_First_Crossing_Probability_Get => My_First_Crossing_Probability
    Excursion_Sets_First_Crossing_Rate_Get => My_First_Crossing_Rate
    Excursion_Sets_Non_Crossing_Rate_Get => My_Non_Crossing_Rate
  end if
  return
end subroutine Method_Initialize
```
where myMethod is the name of this method as will be specified by the excursionSetFirstCrossingMethod input parameter. The procedure pointers Excursion_Sets_First_Crossing_Probability_Get, Excursion_Sets_First_Crossing_Rate_Get, and Excursion_Sets_Non_Crossing_Rate_Get must be set to point functions which return the first crossing probability, first crossing probability rate and noncrossing rate as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The first crossing probability function must have the form:

```fortran
double precision function Excursion_Sets_First_Crossing_Probability(variance,time)

  implicit none
  double precision, intent(in) :: variance, time

  .

  return

end function Excursion_Sets_First_Crossing_Probability
```

The function must return the first crossing probability per unit variance at the specified variance and time.

The first crossing probability rate function must have the form:

```fortran
double precision function Excursion_Sets_First_Crossing_Rate(variance,varianceProgenitor,time)

  implicit none
  double precision, intent(in) :: variance, varianceProgenitor, time

  .

  return

end function Excursion_Sets_First_Crossing_Rate
```

The function must return the rate of first crossing per unit variance at the specified variance and time for a progenitor of the specified varianceProgenitor.

The non-crossing probability rate function must have the form:

```fortran
double precision function Excursion_Sets_First_Non_Crossing_Rate(variance,time)

  implicit none
  double precision, intent(in) :: variance, time

  .

  return

end function Excursion_Sets_First_Non_Crossing_Rate
```

The function must return the rate of trajectories which never cross the barrier at the specified variance and time.

Currently defined excursion set barrier first crossing methods are:

- **linearBarrier** Assures the solution for a linear barrier;
- **Farahi** Solves the first crossing problem using the methodology of Benson et al. [2012];
- **ZhangHui2006** Solves the first crossing problem using the methodology of Zhang and Hui [2006];
- **ZhangHui2006HighOrder** Solves the first crossing problem using a higher order extension of the methodology of Zhang and Hui [2006].
17. Adding New Methods

**Excursion Set Barrier Remapping**

Additional methods for the excursion set barrier can be added using the `excursionSetBarrierRemapInitialize` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `scale` method is described by a directive:

```plaintext
!# <excursionSetBarrierRemapInitialize>
!# <unitName>Excursion_Sets_Barriers_Remap_Scale.Initialize</unitName>
!# </excursionSetBarrierRemapInitialize>
```

Here, `Excursion_Sets_Barriers_Remap_Scale.Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
subroutine Method_Initialize(excursionSetBarrierRemapMethods, barrierName, &
& ratesCalculation, matchedCount)
implicit none
    type(varying_string), intent(in), dimension(:) :: excursionSetBarrierRemapMethods
    type(varying_string), intent(inout) :: barrierName
    logical, intent(in) :: ratesCalculation
    integer, intent(inout) :: matchedCount

    if (any(excursionSetBarrierRemapMethods == 'myMethod')) then
        position = -1
        do i = 1, size(excursionSetBarrierRemapMethods)
            if (excursionSetBarrierRemapMethods(i) == 'myMethod') then
                position = i
                exit
            end if
        end do
        if (ratesCalculation) then
            methodRatesPosition = position
        else
            methodPosition = position
        end if
        matchedCount = matchedCount + 1
        barrierName = barrierName // " :myLabel"
    end if
    return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `excursionSetBarrierRemapMethods` input parameter. The initialization subroutine should identify the position of the matched method in the `excursionSetBarrierRemapMethods()` array and record that it is active for standard barrier calculations (`ratesCalculation` = false) or for barriers used in crossing rate calculations (`ratesCalculation` = true). It should also increment the `matchedCount` argument (to allow checking that all specified barriers were matched) and append a descriptive label to the `barrierName` argument. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The method must provide a subroutine to compute remapping of the barrier as follows:

```plaintext
!# <excursionSetBarrierRemap>
!# <unitName>Method_Barrier_Remap</unitName>
!# </excursionSetBarrierRemap>
```
subroutine Method_Barrier_Remap(barrier, variance, time, ratesCalculation, iRemap)
  implicit none
  double precision, intent(inout) :: barrier
  double precision, intent(in ) :: variance, time
  logical , intent(in ) :: ratesCalculation
  integer , intent(in ) :: iRemap

  if ((ratesCalculation.and.iRemap == methodRatesPosition).or.(.not.ratesCalculation.and.iRemap == methodPosition)) then
    ! Do remapping.
  .
  .
  end if
return
end subroutine Method_Barrier_Remap

and a subroutine to compute remapping of the barrier gradient as follows:

!# <excursionSetBarrierRemapGradient>
!# <unitName>Method_Barrier_Gradient_Remap</unitName>
!# </excursionSetBarrierRemapGradient>
subroutine Method_Barrier_Gradient_Remap(barrier, barrierGradient, variance, time, ratesCalculation, iRemap)
  implicit none
  double precision, intent(inout) :: barrier, barrierGradient
  double precision, intent(in ) :: variance, time
  logical , intent(in ) :: ratesCalculation
  integer , intent(in ) :: iRemap

  if ((ratesCalculation.and.iRemap == methodRatesPosition).or.(.not.ratesCalculation.and.iRemap == methodPosition)) then
    ! Do remapping.
  .
  .
  end if
return
end subroutine Method_Barrier_Gradient_Remap

Currently defined excursion set barrier remapping methods are:

null A null method which leaves the barrier unchanged;
scale Scales the barrier by a multiplicative factor;
Sheth-Mo-Tormen Remaps the barrier according to the algorithm of Sheth et al. [2001].

Galactic Component Radii Solver

Additional methods for solving for radii of galactic components can be added using the galacticStructureRadiusSolverMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the simple method is described by a directive:

!# <galacticStructureRadiusSolverMethod>
!# <unitName>Galactic_Structure_Radii_Simple_Initialize</unitName>
!# </galacticStructureRadiusSolverMethod>
Here, `Galactic_Structure_Radii_Simple_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(galacticStructureRadiusSolverMethod, Galactic_Structure_Radii_Solve_Do)
  implicit none
  type(varying_string), intent(in) :: galacticStructureRadiusSolverMethod
  procedure(), pointer, intent(inout) :: Galactic_Structure_Radii_Solve_Do

  if (galacticStructureRadiusSolverMethod == 'myMethod') Galactic_Structure_Radii_Solve_Do => My_Method_Do_Procedure
  return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `galacticStructureRadiusSolverMethod` input parameter. The procedure pointer `Galactic_Structure_Radii_Solve_Do` must be set to point to a subroutine which solves for the radii of components in a node as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The radii solving subroutine must have the form:

```fortran
subroutine Radii_Solver_Do(thisNode)
  implicit none
  type(treeNode), intent(in), pointer :: thisNode

  .
  .
  return
end subroutine Radii_Solver_Do
```

The function must set the radii (and corresponding circular velocities) of all components that have a radius property in `thisNode`.

Currently defined radius solver methods are:

**simple** This solver computes radii assuming that the gravitational potential is dominated by dark matter (i.e. no baryonic self-gravity is included) and that dark matter does not respond to the presence of baryons (i.e. no adiabatic contraction). It uses the “radius solver” (see §17.4.3) task to interact with the node.

**adiabatic** This solver computes radii including the effects of self-gravity of the baryonic component and adiabatic contraction of the dark matter halo using the method of Gnedin et al. [2004]. It uses the “radius solver” (see §17.4.3) task to interact with the node.

**linear** This solver assumes that radii scale linearly with specific angular momentum, equalling the virial radius when the specific angular momentum equals the product of virial radii and velocities. It uses the “radius solver” (see §17.4.3) task to interact with the node.

**fixed** This solver assumes that radii equal the product of virial radius of the halo and its spin parameter (with an adjustable coefficient). It uses the “radius solver” (see §17.4.3) task to interact with the node.

**Galactic Component Radius Solver Initial Radius**

Additional methods for computing the initial radius in the dark matter profile when solving for adiabatic contraction of the halo can be added using the `galacticStructureRadiusSolverInitialRadiusMethod`
17.4. Existing Method Types

directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the adiabatic method is described by a directive:

```plaintext
!# <galacticStructureRadiusSolverInitialRadiusMethod>
!# <unitName>Galactic_Structure_Initial_Radii_Adiabatic_Initialize</unitName>
!# </galacticStructureRadiusSolverInitialRadiusMethod>
```

Here, `Galactic_Structure_Initial_Radii_Adiabatic_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
subroutine Method_Initialize(galacticStructureRadiusSolverInitialRadiusMethod,Galactic_Structure_Radius_Initial_Get,Galactic_Structure_Radius_Initial_Derivative_Get)
  implicit none
  type(varying_string), intent(in) :: galacticStructureRadiusSolverInitialRadiusMethod
  procedure(), pointer, intent(inout) :: Galactic_Structure_Radius_Initial_Get,Galactic_Structure_Radius_Initial_Derivative_Get
  if (galacticStructureRadiusSolverInitialRadiusMethod == 'myMethod') then
    Galactic_Structure_Radius_Initial_Get => My_Method_Get
    Galactic_Structure_Radius_Initial_Derivative_Get => My_Method_Derivative_Get
  end if
  return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `galacticStructureRadiusSolverInitialRadiusMethod` input parameter. The procedure pointers `Galactic_Structure_Radius_Initial_Get` and `Galactic_Structure_Radius_Initial_Derivative_Get` must be set to point to functions which compute the initial radius in the dark matter halo given the final radius, and the derivative of this quantity with respect to the final radius as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The initial radius function must have the form:

```plaintext
double precision function Radius_Initial_Get(thisNode,radius)
  implicit none
  type (treeNode), intent(in ), pointer :: thisNode
  double precision , intent(in ) :: radius
  return
end function Radius_Initial_Get
```

The function must return the initial radius in the dark matter halo of `thisNode` corresponding to the final radius after accounting for the effects of adiabatic contraction.

The initial radius derivative function must have the form:

```plaintext
double precision function Radius_Initial_Derivative_Get(thisNode,radius)
  implicit none
  type (treeNode), intent(in ), pointer :: thisNode
  double precision , intent(in ) :: radius
  return
end function Radius_Initial_Derivative_Get
```
The function must return the derivative with respect to the final radius of initial radius in the dark matter halo of thisNode corresponding to the final radius after accounting for the effects of adiabatic contraction.

Currently defined initial radius methods are:

**static** This method assumes a static dark matter halo, and so the initial radius always equals the final radius.

**adiabatic** This method assumes adiabatic contraction follows the model of Gnedin et al. [2004].

**Hot Halo Ram Pressure Force**

Additional methods for the ram pressure stripping force due to hot halos can be added using the `hotHaloRamPressureForceMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `Font2008` method is described by a directive:

```
!# <hotHaloRamPressureForceMethod>
!# <unitName>Hot_Halo_Ram_Pressure_Force_Font2008_Initialize</unitName>
!# </hotHaloRamPressureForceMethod>
```

Here, `Hot_Halo_Ram_Pressure_Force_Font2008_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```
subroutine Method_Initialize(hotHaloRamPressureForceMethod,Hot_Halo_Ram_Pressure_Force_Get)
  implicit none
  type(varying_string), intent(in) :: hotHaloRamPressureForceMethod
  procedure(), pointer, intent(inout) :: Hot_Halo_Ram_Pressure_Force_Get

  if (hotHaloRamPressureForceMethod == 'myMethod') Hot_Halo_Ram_Pressure_Force_Get => My_Hot_Halo_Ram_Pressure_Force_Get
  return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `hotHaloRamPressureForceMethod` input parameter. The procedure pointer `Hot_Halo_Ram_Pressure_Force_Get` must be set to point to a function which returns ram pressure force due to the hot halo. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The ram pressure force function must have the form:

```
double precision function Hot_Halo_Ram_Pressure_Force_Get(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode

  .
  .
  .
  return
end function Hot_Halo_Ram_Pressure_Force_Get
```

The function must return the ram pressure force acting on thisNode due to the hot halo of its host node (in units of $M_\odot \text{km}^2 \text{s}^{-1} \text{Mpc}^{-3}$).

Currently defined hot halo ram pressure force methods are:

**null** Returns a zero ram pressure force.

**Font2008** Computes the ram pressure stripping radius using the algorithm of Font et al. [2008].
17.4. Existing Method Types

**Hot Halo Ram Pressure Stripping Radius**

Additional methods for the ram pressure stripping radius in hot halos can be added using the `hotHaloRamPressureStrippingMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `virialRadius` method is described by a directive:

```plaintext
!# <hotHaloRamPressureStrippingMethod>
!# <unitName>Hot_Halo_Ram_Pressure_Stripping_Virial_Radii.Initialize</unitName>
!# </hotHaloRamPressureStrippingMethod>
```

Here, `Hot_Halo_Ram_Pressure_Stripping_Virial_Radii.Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
subroutine Method_Initialize(hotHaloRamPressureStrippingMethod, Hot_Halo_Ram_Pressure_Stripping_Get)
  implicit none
  type(varying_string), intent(in) :: hotHaloRamPressureStrippingMethod
  procedure(), pointer, intent(inout) :: Hot_Halo_Ram_Pressure_Stripping_Get

  if (hotHaloRamPressureStrippingMethod == 'myMethod') Hot_Halo_Ram_Pressure_Stripping_Get => My_Hot_Halo_Ram_Pressure_Stripping_Get
  return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `hotHaloRamPressureStrippingMethod` input parameter. The procedure pointer `Hot_Halo_Ram_Pressure_Stripping_Get` must be set to point to a function which returns the radius to which the hot halo is stripped by ram pressure forces. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The ram pressure stripping radius function must have the form:

```plaintext
double precision function Hot_Halo_Ram_Pressure_Stripping_Get(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  .
  .
  return
end function Hot_Halo_Ram_Pressure_Stripping_Get
```

The function must return the radius (in units of Mpc) to which the hot halo of `thisNode` is stripped by ram pressure forces.

Currently defined hot halo ram pressure stripping radii methods are:

- **virialRadius** Sets the ram pressure stripping radius equal to the virial radius always—effectively resulting in no ram pressure stripping.

- **Font2008** Computes the ram pressure stripping radius using the algorithm of Font et al. [2008].

**Hot Halo Ram Pressure Stripping Timescale**

Additional methods for the ram pressure stripping timescale in hot halos can be added using the `hotHaloRamPressureStrippingTimescaleMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `virialRadius` method is described by a directive:
17. Adding New Methods

Here, \texttt{Hot\_Halo\_Ram\_Pressure\_Timescales\_Halo\_DynTime\_Initialize} is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method\_Initialize(hotHaloRamPressureStrippingTimescaleMethod,Hot\_Halo\_Ram\_Pressure\_Stripping\_Get)
    implicit none
    type(varying\_string), intent(in) :: hotHaloRamPressureStrippingTimescaleMethod
    procedure(), pointer, intent(inout) :: Hot\_Halo\_Ram\_Pressure\_Timescale\_Get

    if (hotHaloRamPressureStrippingTimescaleMethod == 'myMethod') Hot\_Halo\_Ram\_Pressure\_Timescale\_Get => My\_Hot\_Halo\_Ram\_Pressure\_Timescale\_Get
    return
end subroutine Method\_Initialize
```

where \texttt{myMethod} is the name of this method as will be specified by the \texttt{hotHaloRamPressureStrippingTimescaleMethod} input parameter. The procedure pointer \texttt{Hot\_Halo\_Ram\_Pressure\_Timescale\_Get} must be set to point to a function which returns the timescale on which material is removed from the hot halo due to ram pressure forces. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The ram pressure stripping timescale function must have the form:

```fortran
double precision function Hot\_Halo\_Ram\_Pressure\_Timescale\_Get(thisNode)
    implicit none
    type(tree\_Node), intent(inout), pointer :: thisNode

    return
end function Hot\_Halo\_Ram\_Pressure\_Timescale\_Get
```

The function must return the timescale (in units of Gyr) on which the hot halo of \texttt{thisNode} is being stripped by ram pressure forces.

Currently defined hot halo ram pressure stripping timescale methods are:

- \texttt{haloDynamicalTime} Sets the ram pressure stripping timescale equal to the dynamical time of the node’s dark matter halo
- \texttt{ramPressureAcceleration} Computes the timescale from the ram pressure acceleration following Roediger and Brácašggen [2007].

### Hot Halo Temperature Profile

Additional methods for the hot halo temperature profile can be added using the \texttt{hotHaloTemperatureMethod} directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the \texttt{virial} method is described by a directive:

```fortran
!# <hotHaloTemperatureMethod>
!# <unitName>Hot\_Halo\_Temperature\_Virial</unitName>
!# </hotHaloTemperatureMethod>
```

Here, \texttt{Hot\_Halo\_Temperature\_Virial} is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:
17.4. Existing Method Types

```
subroutine Method_Initialize(hotHaloTemperatureMethod, Hot_Halo_Temperature_Get, Hot_Halo_Temperature_Logarithmic_Slope_Get)
  implicit none
  type(varying_string), intent(in) :: hotHaloTemperatureMethod
  procedure(), pointer, intent(inout) :: Hot_Halo_Temperature_Get, Hot_Halo_Temperature_Logarithmic_Slope_Get
  if (hotHaloTemperatureMethod == 'myMethod') then
    Hot_Halo_Temperature_Get => My_Method_Get
    Hot_Halo_Temperature_Logarithmic_Slope_Get => My_Method_Logarithmic_Slope_Get
  end if
  return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `hotHaloTemperatureMethod` input parameter. The procedure pointer `Hot_Halo_Temperature_Get` must be set to point to a function which returns the hot halo density as described below while `Hot_Halo_Temperature_Logarithmic_Slope_Get` must point to a function which returns the logarithmic slope of the temperature profile. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The temperature function must have the form:

```
double precision function Hot_Halo_Temperature_Get(thisNode, radius)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  double precision, intent(in) :: radius
  .
  .
  return
end function Hot_Halo_Temperature_Get
```

The function must return the temperature (in Kelvin) of the hot halo at the specified radius (given in Mpc) for `thisNode`. The logarithmic slope function should have the same template, but return $d \ln T / d \ln r$.

Currently defined hot halo density profile methods are:

- **virial** Implements an isothermal profile with temperature equal to the virial temperature.

**Halo Bias**

Additional methods for the halo bias (i.e. the linear theory bias) can be added using the `darkMatterHaloBiasMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Press-Schechter method is described by a directive:

```
! # <darkMatterHaloBiasMethod>
! # <unitName>Dark_Matter_Halo_Bias_Press_Schechter_Initialize</unitName>
! # </darkMatterHaloBiasMethod>
```

Here, `Dark_Matter_Halo_Bias_Press_Schechter_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```
subroutine Method_Initialize(darkMatterHaloBiasMethod, Dark_Matter_Halo_Bias_Node_Get, Dark_Matter_Halo_Bias_Get)
  implicit none
  type(varying_string), intent(in) :: darkMatterHaloBiasMethod
  procedure(), pointer, intent(inout) :: Dark_Matter_Halo_Bias_Get
```
17. Adding New Methods

if (darkMatterHaloBiasMethod == 'myMethod') then
    Dark_Matter_Halo_Bias_Node_Get => My_Method_Node_Get
    Dark_Matter_Halo_Bias_Get => My_Method_Get
end if
return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the darkMatterHaloBiasMethod input parameter. The procedure pointers Dark_Matter_Halo_Bias_Node_Get and Dark_Matter_Halo_Bias_Get must be set to point to functions which return the bias of the specified halo. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The halo bias functions must have the following forms.

double precision function Dark_Matter_Halo_Bias_Node(thisNode)
    implicit none
    type(treeNode), intent(inout), pointer :: thisNode
    .
    .
    .
    return
end function Dark_Matter_Halo_Bias_Node

The function should return the linear theory bias for thisNode.

double precision function Dark_Matter_Halo_Bias(mass,time)
    implicit none
    double precision, intent(in) :: mass, time
    .
    .
    .
    return
end function Dark_Matter_Halo_Bias

The function should return the linear theory bias for the given mass and time. Two versions of these functions are provided because a common assumption is that the bias depends only on mass and time, while in reality it may depend on other properties of the halo (environment, formation time etc.). The first version of the function allows for arbitrary dependence on properties of the node.

Currently defined halo bias methods are:


SMT Implements the Sheth-Tormen [Sheth et al., 2001] bias.

Tinker2010 Implements the bias described by Tinker et al. [2010].

Halo Mass Functions

Additional methods for the halo mass function can be added using the haloMassFunctionMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Press-Schechter method is described by a directive:
17.4. Existing Method Types

Here, Halo_Mass_Function_Press_Schechter_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
    implicit none
    type(varying_string), intent(in) :: haloMassFunctionMethod
    procedure(), pointer, intent(inout) :: Halo_Mass_Function_Tabulate

    return
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the haloMassFunctionMethod input parameter. The procedure pointer Halo_Mass_Function_Differential_Get must be set to point to a subroutine which returns the differential form of the halo mass function. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The halo mass function function must have the form:

```fortran
double precision function Halo_Mass_Function_Differential_Get(time, mass)
    implicit none
    double precision, intent(in ) :: time, mass

    return
end function Halo_Mass_Function_Differential_Get
```

The function should return the halo mass function, $dn/dM$ (in units of $Mpc^{-3}M_\odot^{-1}$) at mass $mass$ and time $time$.

Currently defined halo mass function methods are:

- **Sheth-Tormen** Implements the Sheth-Tormen [Sheth et al., 2001] mass function.
- **Tinker2008** Implements the mass function described by Tinker et al. [2008].

### Halo Mass Sampling Density Functions

Additional methods for halo mass sampling density functions can be added using the haloMassFunctionSamplingMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the powerLaw method is described by a directive:

```fortran
!# <haloMassFunctionSamplingMethod>
!# <unitName>Merger_Trees_Mass_Function_Sampling_Power_Law_Initialize</unitName>
!# </haloMassFunctionSamplingMethod>
```

Here, Merger_Trees_Mass_Function_Sampling_Power_Law_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:
17. Adding New Methods

```fortran
subroutine Method_Initialize(haloMassFunctionSamplingMethod, Merger_Tree_Construct_Mass_Function_Sampling_Get)
    implicit none
    type(varying_string), intent(in) :: haloMassFunctionSamplingMethod
    procedure(), pointer, intent(inout) :: Merger_Tree_Construct_Mass_Function_Sampling_Get

    if (haloMassFunctionSamplingMethod == 'myMethod') Merger_Tree_Construct_Mass_Function_Sampling_Get => My_Mass_Function_Sampling
    return
end subroutine Method_Initialize
```

where **myMethod** is the name of this method as will be specified by the **haloMassFunctionSamplingMethod** input parameter. The procedure pointer **Merger_Tree_Construct_Mass_Function_Sampling_Get** must be set to point to a function which returns the sampling rate per unit decade of halo mass.

The halo mass sampling density function must have the form:

```fortran
double precision function My_Mass_Function_Sampling(mass, time, massMinimum, massMaximum)
    implicit none
    double precision, intent(in) :: mass, time, massMinimum, massMaximum
    .
    .
    return
end function My_Mass_Function_Sampling
```

The function should return the halo mass sampling density function (the relative number of halos per decade of halo mass to sample) for halos of the given **mass**. Halos are defined at the given **time** and will be sampled in the mass range **massMinimum** to **massMaximum**.

Currently defined halo mass sampling density function methods are:

- **powerLaw** The distribution of halo masses is such that the mass of the i<sup>th</sup> halo is
  \[ M_{\text{halo},i} = \exp \left[ \ln(M_{\text{halo, min}}) + \ln \left( \frac{M_{\text{halo, max}}}{M_{\text{halo, min}}} \right) x_i \right]^{1+\alpha} \].
  (17.6)

  Here, \( x_i \) is a number between 0 and 1 and \( \alpha = \text{mergerTreeBuildTreesHaloMassExponent} \) is an input parameter that controls the relative number of low and high mass tree produced.

- **haloMassFunction** The sampling density is set equal to the dark matter halo mass function, defined per decade of halo mass.

- **stellarMassFunction** The sampling density is chosen to give optimally minimal errors on the model stellar mass function (see §10.4.1 for full details).

**Halo Spin Distribution**

Additional methods for the halo spin distribution can be added using the **haloSpinDistributionMethod** directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the **lognormal** method is described by a directive:

```
!# <haloSpinDistributionMethod>
!# <unitName>Halo_Spin_Distribution_Lognormal_Initialize</unitName>
!# </haloSpinDistributionMethod>
```

Here, **Halo_Spin_Distribution_Lognormal_Initialize** is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:
subroutine Method_Initialize(haloSpinDistributionMethod,Halo_Spin_Sample_Get)
  implicit none
  type(varying_string), intent(in) :: haloSpinDistributionMethod
  procedure(), pointer, intent(inout) :: Halo_Spin_Sample_Get

  if (haloSpinDistributionMethod == 'myMethod') Halo_Spin_Sample_Get => My_Method_Get
  return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the haloSpinDistributionMethod input parameter. The procedure pointer Halo_Spin_Sample_Get must be set to point to a function which returns a halo spin drawn at random from the distribution. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The halo spin sampling function must have the form:

double precision function Halo_Spin_Sample_Get(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  .
  .
  .
  return
end function Halo_Spin_Sample_Get

The function must return a halo spin drawn at random for the distribution appropriate to thisNode.

Currently defined halo spin distribution methods are:

lognormal  Implements a lognormal distribution with median lognormalSpinDistributionMedian and dispersion in ln λ of lognormalSpinDistributionSigma, both of which are input parameters to GALACTICUS.

Bett2007  Implements distribution from Bett et al. [2007] with parameter λ₀ = [spinDistributionBett2007Lambda0] and α = [spinDistributionBett2007Alpha], both of which are input parameters to GALACTICUS.

Initial Mass Function Functions

Each registered IMF must provide multiple functions, specified by the following directives:

!# <imfRecycledInstantaneous>
!# <unitName>Star_Formation_IMF_Recycled_Instantaneous_My_IMF</unitName>
!# </imfRecycledInstantaneous>

!# <imfYieldInstantaneous>
!# <unitName>Star_Formation_IMF_Yield_Instantaneous_My_IMF</unitName>
!# </imfYieldInstantaneous>

!# <imfTabulate>
!# <unitName>Star_Formation_IMF_Tabulate_My_IMF</unitName>
!# </imfTabulate>

!# <imfMinimumMass>
!# <unitName>Star_Formation_IMF_Minimum_Mass_My_IMF</unitName>

801
17. Adding New Methods

These functions/subroutines should have the following forms:

```fortran
subroutine Star_Formation_IMF_Recycled_Instantaneous_My_IMF(imfSelected, imfMatched, recycledFraction)
  integer, intent(in) :: imfSelected
  logical, intent(inout) :: imfMatched
  double precision, intent(out) :: recycledFraction

  if (imfSelected == imfIndex) then
    imfMatched=.true.
  end if
  return
end subroutine Star_Formation_IMF_Recycled_Instantaneous_My_IMF

subroutine Star_Formation_IMF_Yield_Instantaneous_My_IMF(imfSelected, imfMatched, yield)
  integer, intent(in) :: imfSelected
  logical, intent(inout) :: imfMatched
  double precision, intent(out) :: yield

  if (imfSelected == imfIndex) then
    imfMatched=.true.
  end if
  return
end subroutine Star_Formation_IMF_Yield_Instantaneous_My_IMF

subroutine Star_Formation_IMF_Tabulate_My_IMF(imfSelected, imfMatched, imfMass, imfPhi)
  integer, intent(in) :: imfSelected
  logical, intent(inout) :: imfMatched
  double precision, intent(inout), allocatable, dimension(:) :: imfMass, imfPhi

  if (imfSelected == imfIndex) then
    imfMatched=.true.
  end if
```

802
return
end subroutine Star_Formation_IMF_Tabulate_My_IMF

subroutine Star_Formation_IMF_Minimum_Mass_My_IMF(imfSelected,imfMatched,minimumMass)
implicit none
integer, intent(in) :: imfSelected
logical, intent(inout) :: imfMatched
double precision, intent(out) :: minimumMass

if (imfSelected == imfIndex) then
    .
    .
    .
    imfMatched=.true.
end if
return
end subroutine Star_Formation_IMF_Minimum_Mass_My_IMF

subroutine Star_Formation_IMF_Maximum_Mass_My_IMF(imfSelected,imfMatched,minimumMass)
implicit none
integer, intent(in) :: imfSelected
logical, intent(inout) :: imfMatched
double precision, intent(out) :: maximumMass

if (imfSelected == imfIndex) then
    .
    .
    .
    imfMatched=.true.
end if
return
end subroutine Star_Formation_IMF_Maximum_Mass_My_IMF

subroutine Star_Formation_IMF_Phi_My_IMF(imfSelected,imfMatched,imfMass,imfPhi)
integer, intent(in) :: imfSelected:: imfSelected
logical, intent(inout) :: imfMatched:: imfMatched
double precision, intent(inout) :: imfMass:: imfMass
double precision, intent(out) :: imfPhi:: imfPhi

if (imfSelected == imfIndex) then
    .
    .
    .
    imfMatched=.true.
end if
return
end subroutine Star_Formation_IMF_Phi_My_IMF

In each case the procedure should check if the supplied imfSelected index matches the index which this IMF was given when it was registered. If it is, then imfMatched should be set to true. The procedures
should then perform as follows:

*Star* Formation_\text{IMF} Yield\_Instantaneous\_My\_IMF* Return a suitable metal yield in \textit{yield} for this IMF in the instantaneous recycling approximation.

*Star* Formation_\text{IMF} Recycled\_Instantaneous\_My\_IMF* Return a suitable recycled fraction in \textit{recycledFraction} for this IMF in the instantaneous recycling approximation.

*Star* Formation_\text{IMF} Tabulate\_My\_IMF* Allocate the \textit{imfMass()} and \textit{imfPhi()} arrays and fill them with a tabulation of the IMF. The routine can choose the size of the tabulation and should ensure that it is sufficient to resolve any features in the IMF.

*Star* Formation_\text{IMF} Minimum\_Mass\_My\_IMF* Return the lowest mass for which the IMF is non-zero.

*Star* Formation_\text{IMF} Maximum\_Mass\_My\_IMF* Return the largest mass for which the IMF is non-zero.

*Star* Formation_\text{IMF} Phi\_My\_IMF* Return the IMF for the specified \textit{imfMass} initial stellar mass.

Currently defined IMFs are described in §13.22.2.

**Initial Mass Function Selection**

Additional methods for selection of initial mass functions can be added using the \texttt{imfSelectionMethod} directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the \texttt{fixed} method is described by a directive:

```bash
!# <imfSelectionMethod>
!# <unitName>IMF_Select_Fixed.Initialize</unitName>
!# </imfSelectionMethod>
```

Here, \texttt{IMF_Select\_Fixed\_Initialize} is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method\_Initialize(imfSelectionMethod,IMF\_Select,imfNames)
  implicit none
  type(varying_string), intent(in) :: imfSelectionMethod,imfNames(:)
  procedure(), pointer, intent(inout) :: IMF\_Select

  if (imfSelectionMethod == 'myMethod') then
    IMF\_Select\_Fixed => My\_Selection\_Procedure
  end if
  return
end subroutine Method\_Initialize
```

where \texttt{myMethod} is the name of this method as will be specified by the \texttt{imfSelectionMethod} input parameter. The procedure pointer \texttt{IMF\_Select} must be set to point to a function which returns the index of the selected IMF as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.). The input array \texttt{imfNames()} contains a list of all available IMF names and can be used for index determination.

The selection function must have the form:
integer function IMF_Select(starFormationRate, fuelAbundances, component)
  double precision, intent(in) :: starFormationRate
  type(abundancesStructure), intent(in) :: fuelAbundances
  integer, intent(in) :: component
  
  return
end function IMF_Select

The function must return the index of the IMF appropriate for the given starFormationRate (in $M_\odot$ Gyr$^{-1}$), fuelAbundances and component (using the component labels provided by the Galactic_Structure_Options module).

Currently defined IMF selection methods are:

- **fixed**: A fixed IMF is used irrespective of physical conditions. The IMF is specified by the input parameter imfSelectionFixed.

- **diskSpheroid**: Uses different IMFs for star formation in disks and in spheroids irrespective of other physical conditions. The IMFs are specified by the input parameters imfSelectionDisk and imfSelectionSpheroid.

### Linear Growth Function

Additional methods for the linear growth factor can be added using the linearGrowthMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the simple method is described by a directive:

```fortran
!# <linearGrowthMethod>
!# <unitName>Growth_Factor_Simple_Initialize</unitName>
!# </linearGrowthMethod>
```

Here, `Growth_Factor_Simple_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(linearGrowthMethod, Linear_Growth_Tabulate)
  implicit none
  type(varying_string), intent(in) :: linearGrowthMethod
  procedure(), pointer, intent(inout) :: Linear_Growth_Tabulate

  if (linearGrowthMethod.eq.'myMethod') then
    Linear_Growth_Tabulate => My_Do_Tabulate
  
  end if
  
  return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the linearGrowthMethod input parameter. The procedure pointer `Linear_Growth_Tabulate` must be set to point to a subroutine which tabulates the linear growth factor as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The tabulation subroutine must have the form:
subroutine Linear_Growth_Tabulate(time, growthTableNumberPoints, growthTableTime, growthTableWavenumber, & growthTableGrowthFactor, normalizationMatterDominated)
    implicit none
    double precision, intent(in) :: time
    integer, intent(out) :: growthTableNumberPoints
    double precision, intent(inout), allocatable, dimension(:) :: growthTableTime, growthTableWavenumber
    double precision, intent(inout), allocatable, dimension(:, :) :: growthTableGrowthFactor
    double precision, intent(out), dimension(3) :: normalizationMatterDominated
    return
end subroutine Linear_Growth_Tabulate

The subroutine must tabulate the linear growth factor in array growthTableGrowthFactor() for dark matter, baryons and radiation (entries 1, 2 and 3 of the first dimension respectively) as a function of time growthTableTime() (second dimension) and wavenumber growthTableWavenumber() (third dimension). These arrays must be allocated to the correct size, and may be previously allocated, therefore requiring a deallocation. The number of tabulated points in the time dimension should be returned in growthTableNumberPoints. It is permissible to tabulate for just a single wavenumber if the growth function is independent of wavenumber. The subroutine should ensure that the currently requested time is within the range of the tabulated function (preferably with some buffer). The linear growth factors must be normalized to unity at $a = 1$. Additionally, normalizationMatterDominated should be set to the factor by which the tabulated growth factor (for the smallest wavenumber tabulated) must be multiplied such that it scales as $(9\Omega_M/4\theta_0)_{1/3}(H_0t)^{2/3}$ during the matter dominated regime.

Currently defined linear growth factor methods are:

**simple** The linear growth factor is computed for a Universe containing collisionless matter and a cosmological constant.

**Merger Tree Branching**

Additional methods for merger tree branching can be added using the treeBranchingMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the modifiedPress-Schechter method is described by a directive:

```
!# <treeBranchingMethod>
!# <unitName>Modified_Press_Schechter_Branching.Initialize</unitName>
!# </treeBranchingMethod>
```

Here, Modified_Press_Schechter_Branching.Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```subroutine Method.Initialize(treeBranchingMethod, TreeBranchingProbability, TreeSubresolutionFraction, TreeBranch_Mass, TreeMaximumStep)
    type(varying_string), intent(in) :: treeBranchingMethod
    procedure(), pointer, intent(inout) :: TreeBranchingProbability, TreeSubresolutionFraction, TreeBranch_Mass, TreeMaximumStep
    if (treeBranchingMethod == 'myMethod') then
        TreeBranchingProbability => My_Branching_Probability
        TreeSubresolutionFraction => My_Subresolution_Fraction
        TreeMaximumStep => My_Maximum_Step
        TreeBranch_Mass => My_Branch_Mass
    .
```

806
17.4. Existing Method Types

where method is the name of this method as will be specified by the treeBranchingMethod input parameter. The procedure pointers must be set to point to routines which perform various functions as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The procedure pointers must point to functions with the following templates:

```fortran
double precision function My_Branch_Mass(haloMass,deltaCritical,massResolution,probability)
    double precision, intent(in) :: haloMass,deltaCritical,massResolution,probability
    .
    .
    return
end function My_Branch_Mass
```

```fortran
double precision function My_Branching_Maximum_Step(haloMass,deltaCritical,massResolution)
    double precision, intent(in) :: haloMass,deltaCritical,massResolution
    .
    .
    return
end function My_Branching_Maximum_Step
```

```fortran
double precision function My_Branching_Probability(haloMass,deltaCritical,massResolution)
    double precision, intent(in) :: haloMass,deltaCritical,massResolution
    .
    .
    return
end function My_Branching_Probability
```

```fortran
double precision function My_Subresolution_Fraction(haloMass,deltaCritical,massResolution)
    double precision, intent(in) :: haloMass,deltaCritical,massResolution
    .
    .
    return
end function My_Subresolution_Fraction
```

Tree_Branching_Probability must point to a function which returns the probability per unit change in \( \delta_{\text{crit}} \) that a halo of mass haloMass at time deltaCritical will undergo a branching to progenitors with mass greater than massResolution. Tree_Subresolution_Fraction must point to a function which returns the fraction of mass accreted in subresolution halos, i.e. those below massResolution, per unit change in \( \delta_{\text{crit}} \) for a halo of mass haloMass at time deltaCritical, or a negative value if the halo is so close to the resolution limit that this number cannot be determined accurately. Tree_Maximum_Step must point to a function which returns the maximum allowed step in \( \delta_{\text{crit}} \) that a halo of mass haloMass
17. Adding New Methods

At time $\Delta_{\text{critical}}$ should be allowed to take. $\text{Tree\_Branch\_Mass}$ must point to a function which returns the mass of one of the halos to which the given halo branches, given the branching probability, probability.

Currently defined merger tree branching methods are:

- **modifiedPress-Schechter** Branching probabilities are computed using the method of Parkinson et al. [2008]. Progenitor mass functions generated using GALACTICUS’s implementation of this algorithm (and the Cole et al. [2000] merger tree building algorithm) are shown in Fig. 17.2.

- **generalizedPress-Schechter** Branching probabilities are computed using excursion set barrier first crossing rates (computed using the selected $\text{excursionSetFirstCrossingMethod}$; see §17.4.1), modified by the selected $\text{treeBranchingModifierMethod}$ (see §17.4.1).

### Merger Tree Branching Modifiers

Additional methods for merger tree branching probability modifiers can be added using the $\text{treeBranchingModifierMethod}$ directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the null method is described by a directive:

```verbatim
#!/ <treeBranchingModifierMethod>
#!/ <unitName>Merger_Tree_Branching_Modifiers_Null_Initialize</unitName>
#!/ </treeBranchingModifierMethod>
```

Here, $\text{Merger\_Tree\_Branching\_Modifiers\_Null\_Initialize}$ is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(treeBranchingModifierMethod, Merger_Tree_Branching_Modifier_Get)
  type (varying_string), intent(in) :: treeBranchingModifierMethod
  procedure(double precision), pointer, intent(inout) :: Merger_Tree_Branching_Modifier_Get

  if (treeBranchingModifierMethod == 'myMethod') then
    Merger_Tree_Branching_Modifier_Get => My_Get
  end if
  return
end subroutine Method_Initialize
```

where $\text{myMethod}$ is the name of this method as will be specified by the $\text{treeBranchingModifierMethod}$ input parameter. The procedure pointer $\text{Merger\_Tree\_Branching\_Modifier\_Get}$ must build and return the modifier to the branching probability as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The branching probability modifier function should have the interface:

```fortran
subroutine Merger_Tree_Branching_Modifier_Get(parentDelta, childSigma, parentSigma)
  double precision, intent(in) :: parentDelta, childSigma, parentSigma

  return
end subroutine Merger_Tree_Branching_Modifier_Get
```
Figure 17.2.: Progenitor mass functions at redshifts $z = 0.5, 1, 2$ and $4$ (bottom to top) for halos of mass $10^{12\pm0.151}, 10^{13.5\pm0.151}$ and $10^{15\pm0.151} h^{-1} M_\odot$ (left to right) are shown. Green lines are measured from the Millennium Simulation, while red lines are computed using GALACTICUS’s merger tree building routines (with the Parkinson et al. [2008] branching algorithm and the Cole et al. [2000] tree building algorithm).
and should return the multiplicative modifier to the branching probability for the given parentDelta, childSigma and parentSigma.

Currently defined merger tree branching probability modifier methods are:

null Makes no modification;

Parkinson-Cole-Helly2008 Modifies branching rates according to the algorithm of Parkinson et al. [2008].

Merger Tree Building

Additional methods for merger tree building can be added using the mergerTreeBuildMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Cole2000 method is described by a directive:

```mergerTreeBuildMethod
<unitName>Merger_Tree_Build_Cole2000_Initialize</unitName>
</mergerTreeBuildMethod>
```

Here, Merger_Tree_Build_Cole2000_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(mergerTreeBuildMethod,Merger_Tree_Build)
  type(varying_string), intent(in) :: mergerTreeBuildMethod
  procedure(), pointer, intent(inout) :: Merger_Tree_Build
  if (mergerTreeBuildMethod == 'myMethod') then
    Merger_Tree_Build => My_Do_Tabulate
  .
  .
  end if
  return
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the mergerTreeBuildMethod input parameter. The procedure pointer Merger_Tree_Build must build and return a merger tree given the a base node as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

```fortran
subroutine Merger_Tree_Build_Do(thisTree)
  type(mergerTree), intent(inout) :: thisTree
  .
  .
  return
end subroutine Merger_Tree_Build_Do
```

and should return a full merger tree in thisTree built from the base node which will already be set in thisTree. The tree must have at least masses, times and parent/child/sibling links created. Other properties (e.g. spins) can be optionally included also.

Currently defined merger tree building methods are:

17.4. Existing Method Types

**Merger Tree Building Mass Resolution**

Additional methods for the mass resolution to use when building merger trees can be added using the `mergerTreesBuildMassResolution` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `fixed` method is described by a directive:

```plaintext
!# <mergerTreesBuildMassResolution>
!# <unitName>Merger_Trees_Build_Mass_Resolution_Fixed_Initialize</unitName>
!# </mergerTreesBuildMassResolution>
```

Here, `Merger_Trees_Build_Mass_Resolution_Fixed_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
subroutine Method_Initialize(mergerTreesBuildMassResolutionMetod,Merger_Tree_Build_Mass_Resolution_Get)
  type(varying_string), intent(in) :: mergerTreesBuildMassResolutionMetod
  procedure(), pointer, intent(inout) :: Merger_Tree_Build_Mass_Resolution_Get
  if (mergerTreesBuildMassResolutionMetod == 'myMethod') then
    Merger_Tree_Build_Mass_Resolution_Get => My_Merger_Tree_Build_Mass_Resolution
  .
  .
  end if
  return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `mergerTreesBuildMassResolutionMetod` input parameter. The procedure pointer `Merger_Tree_Build_Mass_Resolution_Get` must be set to point to a function which returns the mass resolution to use for a given tree as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The mass resolution function should have the form:

```plaintext
double precision function Merger_Tree_Build_Mass_Resolution(thisTree)
  type(mergerTree), intent(in ) :: thisTree
  .
  .
  return
end function Merger_Tree_Build_Mass_Resolution
```

and should return the mass resolution to use when building `thisTree`. Note that the base node in `thisTree` will already be set.

Currently defined merger tree building mass resolution methods are:

- **fixed** Uses a fixed mass resolution for all trees.
- **scaled** The resolution is set to a fraction of the tree base node mass (or a defined minimum, whichever is larger).
17. Adding New Methods

Merger Tree Construction

Additional methods for merger tree construction can be added using the `mergerTreeConstructMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `build` method is described by a directive:

```mergerTreeConstructMethod
!# <mergerTreeConstructMethod>
!# <unitName>Merger_Tree_Build.Initialize</unitName>
!# </mergerTreeConstructMethod>
```

Here, `Merger_Tree_Build.Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(mergerTreeConstructMethod,Merger_Tree_Construct)
  type(varying_string), intent(in) :: mergerTreeConstructMethod
  procedure(), pointer, intent(inout) :: Merger_Tree_Construct
  if (mergerTreeConstructMethod == 'myMethod') then
    Merger_Tree_Construct => My_Do_Tabulate
    ...
  end if
  return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `mergerTreeConstructMethod` input parameter. The procedure pointer `Merger_Tree_Construct` must be set to point to a function which returns a fully constructed merger tree as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The construction subroutine should have the following form:

```fortran
subroutine Merger_Tree_Construct_Do(thisTree,skipTree)
  type(mergerTree), intent(inout) :: thisTree
  logical, intent(in) :: skipTree
  return
end subroutine Merger_Tree_Construct_Do
```

and should return a full merger tree in `thisTree`, unless `skipTree` is true, in which case this tree will be skipped (i.e. not evolved or output) and so it suffices to merely allocate the base node—there is no need to create the entire tree (although it is permissible to do so)—and update any internal data (e.g. a count of trees constructed) as required. The tree must have at least masses, times and parent/child/sibling links created. Other properties (e.g. spins) can be optionally included also. By default, the tree is assumed to be “uninitialized”, such that the merger tree initialization function will be called prior to the tree being evolve. If the tree construction method returns a fully initialized tree it should set `thisTree%initialized=.true.`.

Currently defined merger tree construction methods are:

- **build** Generates a set of halo masses distributed between `mergerTreeBuildHaloMassMinimum` and `mergerTreeBuildHaloMassMaximum` (with `mergerTreeBuildTreesPerDecade` halos per decade of mass) at redshift `mergerTreeBuildTreesBaseRedshift`. 
or with masses read from a file, and then uses the selected merger tree build method (see §17.4.1) to build trees from these base nodes;

**read**  Reads merger tree data from an HDF5 file (see §6.2). The file to read is specified by the [mergerTreeReadFileName] parameter.

**smoothAccretion** Constructs a branchless merger tree with a smooth accretion history using the selected mass accretion history method (see §17.4.1). See §13.25.4 for details.

**stateRestore** Intended primarily for debugging purposes, this method will restore a tree whose complete internal state was written to file. See §13.25.5 for details of how to use this method.

**fullySpecified** Intended primarily for constructing test cases, this method allows the full state of the merger tree (and all components of nodes) to be specified via an XML document. See §13.25.3 for details of how to use this method.

### Non-linear Matter Power Spectrum

Additional methods for the non-linear matter power spectrum can be added using the `powerSpectrumNonlinearMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Peacock-Dodds1996 method is described by a directive:

```
!# <powerSpectrumNonlinearMethod>
!# <unitName>Nonlinear_Power_Spectrum_Power_Law.Initialize</unitName>
!# </powerSpectrumNonlinearMethod>
```

Here, `Power_Spectrum_Nonlinear_PeacockDodds1996.Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(powerSpectrumNonlinearMethod,Power_Spectrum_Nonlinear_Get)
  implicit none
  type (varying_string ), intent(in ) :: powerSpectrumNonlinearMethod
  procedure(double precision), pointer, intent(inout) :: Power_Spectrum_Nonlinear_Get

  if (powerSpectrumNonlinearMethod == 'myMethod') then
    Power_Spectrum_Nonlinear_Get => My_Get
  .
  .
  end if
  return
end subroutine Method_Initialize
```

where **myMethod** is the name of this method as will be specified by the `powerSpectrumNonlinearMethod` input parameter. The procedure pointer `Power_Spectrum_Nonlinear_Get` must be set to point to a subroutine which computes the non-linear matter power spectrum as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The non-linear matter power spectrum function must have the form:

```fortran
double precision function Nonlinear_Power_Spectrum(wavenumber,time)
  implicit none
  double precision, intent(in ) :: wavenumber,time
```

813
17. Adding New Methods

The function must return the non-linear matter power spectrum, \( P_n(k) \), at the requested wavenumber (in units of Mpc\(^{-1}\)) and time (in units of Gyr).

Currently defined non-linear matter power spectrum methods are:

- **linear** Simply returns the linear matter power spectrum. Intended primarily for testing purposes.
- **Peacock-Dodds1996** Uses the fitting function of Peacock and Dodds [1996] to compute the non-linear matter power spectrum.
- **CosmicEmu** Utilizes the cosmic emulator (“CosmicEmu”) code of Lawrence et al. [2010] to compute the non-linear matter power spectrum.

### Chemical Reaction Rates

Additional methods for chemical species reaction rates can be added using the `chemicalReactionRatesMethods` directive. Note that more than one method can be specified in which cases rates are cumulative over all selected methods. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `hydrogenNetwork` method is described by a directive:

```plaintext
!# <chemicalReactionRatesMethods>
!# <unitName>Chemical_Hydrogen_Rates_Initialize</unitName>
!# </chemicalReactionRatesMethods>
```

Here, `Chemical_Hydrogen_Rates_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
subroutine Method_Initialize(chemicalReactionRatesMethods)
    implicit none
    type(varying_string), intent(in) :: chemicalReactionRatesMethods
    if (chemicalReactionRatesMethods == 'myMethod') then
        ratesSelected = .true.
    .
    .
    end if
    return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `chemicalReactionRatesMethods` input parameter. The `ratesSelected` variable is set to true if the method is active and will be checked on all subsequent calls to the module such that rates are computed only if `ratesSelected` is true. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The method must provide a subroutine to compute the chemical reaction rates. This subroutine is specified by the `chemicalRatesCompute` directive. The directive should contain a single argument, giving the name of a subroutine to be called to compute rates. For example, the `hydrogenNetwork` method uses:
Here, *Chemical_Hydrogen_Rates_Compute* is the name of a subroutine which will be called to compute the rates. The rates subroutine must have the following form:

```fortran
subroutine Compute_Rates(temperature, chemicalDensity, radiation, chemicalRates)
  implicit none
  type(chemicalAbundancesStructure), intent(in) :: chemicalDensity
  double precision, intent(in) :: temperature
  type(radiationStructure), intent(in) :: radiation
  type(chemicalAbundancesStructure), intent(inout) :: chemicalRates

  ! Exit immediately if this method is not active.
  if (.not.ratesSelected) return

  ! Compute rates for all species present.
  .
  .
  .
  return
end subroutine Compute_Rates
```

Here, *temperature* is the temperature of the gas, *chemicalDensity* provides the densities (in cm\(^{-3}\)) of all chemicals, the radiation field is described by the *radiation* object and any reaction rates should be added to the *chemicalRates* object in units of cm\(^{-3}\) s\(^{-1}\).

Currently defined chemical reaction rate methods are:

- **null** A null method which does not affect any rates.
- **hydrogenNetwork** Computes rates using the network of reactions and fitting functions from Abel et al. [1997] and Tegmark et al. [1997].

**Population III Supernovae**

Additional methods for Population III supernovae can be added using the *supernovaePopIIIMethod* directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the *Heger-Woosley2002* method is described by a directive:

```fortran
! <supernovaePopIIIMethod>
! <unitName>Supernovae_Population_III_HegerWoosley_Initialize</unitName>
! </supernovaePopIIIMethod>
```

Here, *Supernovae_Population_III_HegerWoosley_Initialize* is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(supernovaePopIIIMethod, SNePopIII_Cumulative_Energy_Get)
  implicit none
  type(varying_string), intent(in) :: supernovaePopIIIMethod
  procedure(), pointer, intent(inout) :: SNePopIII_Cumulative_Energy_Get

  if (supernovaePopIIIMethod == 'myMethod') then
```
17. Adding New Methods

```fortran
SNePopIII_Cumulative_Energy_Get => My_SNePopIII_Cumulative_Energy_Get

end if
return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `supernovaePopIIIMethod` input parameter. The procedure pointer `SNePopIII_Cumulative_Energy_Get` must be set to point to a function which returns the cumulative energy input from Population III supernovae as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The functions must have the form:

```fortran
double precision function PopIII_Cumulative_Energy(initialMass, age, metallicity)
imPLICIT none
  double precision, intent(in) :: initialMass, age, metallicity
.
.
return
end function PopIII_Cumulative_Energy
```

This function must return the cumulative energy (in $M_\odot (\text{km/s})^2$) from Population III supernovae resulting from a star with given `initialMass` and `metallicity` after a time `age`.

Currently defined population III supernovae methods are:

**Heger-Woosley2002** Computes the energy input from the pair-instability results of Heger and Woosley [2002].

**Power Spectrum Variance Window Function**

Additional methods for the window function used to compute variance from the power spectrum can be added using the `powerSpectrumWindowFunctionMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `topHat` method is described by a directive:

```fortran
! # <powerSpectrumWindowFunctionMethod>
! # <unitName>Power_Spectrum_Window_Functions_Top_Hat_Initialize</unitName>
! # </powerSpectrumWindowFunctionMethod>
```

Here, `Power_Spectrum_Window_Functions_Top_Hat_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(powerSpectrumWindowFunctionMethod, Power_Spectrum_Window_Function_Get)
imPLICIT none
  type (varying_string ), intent(in ) :: powerSpectrumWindowFunctionMethod
  procedure(double precision), pointer, intent(inout) :: Power_Spectrum_Window_Function_Get
  procedure(double precision), pointer, intent(inout) :: Power_Spectrum_Window_Function_Wavenumber_Maximum_Get
if (powerSpectrumWindowFunctionMethod == 'myMethod') then
  Power_Spectrum_Window_Function_Get => My_Window_Function
end if
return
end subroutine Method_Initialize
```

Heger-Woosley2002 Computes the energy input from the pair-instability results of Heger and Woosley [2002].
17.4. Existing Method Types

Power_Spectrum_Window_Function_Wavenumber_Maximum_Get => My_Window_Function_Maximum_Wavelength

where myMethod is the name of this method as will be specified by the powerSpectrumWindowFunctionMethod input parameter. The procedure pointers Power_Spectrum_Window_Function_Get, and Power_Spectrum_Window_Function_Wavenumber_Maximum_Get must be set to point to functions which return the window function for a given wavenumber and mass, and the maximum wavenumber for which that window function is non-zero respectively. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The window function function must have the form:

subroutine Power_Spectrum_Window_Function(wavenumber,smoothingMass)
  implicit none
  double precision, intent(in ) :: wavenumber,smoothingMass
.
.
return
end subroutine Power_Spectrum_Window_Function

The function should return the window function for the specified wavenumber (given in Mpc$^{-1}$) and the given smoothingMass (given in $M_\odot$).

The window function maximum wavenumber function must have the form:

subroutine Power_Spectrum_Window_Function_Wavenumber_Maximum(smoothingMass)
  implicit none
  double precision, intent(in ) :: smoothingMass
.
.
return
end subroutine Power_Spectrum_Window_Function_Wavenumber_Maximum

The function should return the largest wavenumber for which the window function is non-zero for the given smoothingMass (given in $M_\odot$). If the window function is non-zero as $k \to \infty$ then a suitably large value (e.g. $10^{30}\text{Mpc}^{-1}$) should be returned.

Currently defined power spectrum variance window function methods are:

topHat The window function is a top-hat in real-space.

kSpaceSharp The window function is a top-hat in $k$-space.

topHatKSpaceSharpHybrid A convolution of top-hat in real space and top-hat in $k$-space window functions.
### Primordial Power Spectrum

Additional methods for the primordial power spectrum can be added using the `powerSpectrumMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `powerLaw` method is described by a directive:

```plaintext
!# <powerSpectrumMethod>
!# <unitName>CDM_Primordial_Power_Spectrum_Power_Law_Initialize</unitName>
!# </powerSpectrumMethod>
```

Here, `CDM_Primordial_Power_Spectrum_Power_Law_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(powerSpectrumMethod, Power_Spectrum_Tabulate)
  implicit none
  type(varying_string), intent(in) :: powerSpectrumMethod
  procedure(), pointer, intent(inout) :: Power_Spectrum_Tabulate

  if (powerSpectrumMethod.eq.'myMethod') then
    Power_Spectrum_Tabulate => My_Do_Tabulate
  end if
  return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `powerSpectrumMethod` input parameter. The procedure pointer `Power_Spectrum_Tabulate` must be set to point to a subroutine which tabulates the power spectrum as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The tabulation subroutine must have the form:

```fortran
subroutine Power_Spectrum_Tabulate(wavenumber, powerSpectrumNumberPoints, powerSpectrumLogWavenumber, powerSpectrumLogP)
  implicit none
  double precision, intent(in) :: wavenumber
  double precision, allocatable, dimension(:), intent(inout) :: powerSpectrumLogWavenumber, powerSpectrumLogP
  integer, intent(out) :: powerSpectrumNumberPoints

  return
end subroutine Power_Spectrum_Tabulate
```

The subroutine must tabulate the natural log of the power spectrum in array `powerSpectrumLogP()` as a function of the natural log of wavenumber `powerSpectrumLogWavenumber()` (these arrays must be allocated to the correct size, and may be previously allocated, therefore requiring a deallocation). The number of tabulated points should be returned in `powerSpectrumNumberPoints`. The subroutine should ensure that the currently requested `wavenumber` is within the range of the tabulated function (preferably with some buffer).

Currently defined power spectrum methods are:
17.4. Existing Method Types

powerLaw The power spectrum is assumed to be a power law, possibly with a running index. It is defined by

\[ P(k) \propto k^{n_s + \ln(k/k_{ref})/dn/d\ln k}, \]

where the parameters are specified by input parameters \( n_s \equiv \text{power Spectrum Index}, k_{ref} \equiv \text{power Spectrum Reference Wavenumber} \) and \( dn/d\ln k \equiv \text{power Spectrum Running} \).

Radiation Components

Radiation components (i.e. types of radiation field that may be added to any radiation object; see §17.5.3) are defined using a combination of several directives: \texttt{radiationLabel, radiationSet, radiationTemperature} and \texttt{radiationFlux}. For example, the cosmic microwave background radiation component is defined by the following set of directives:

```plaintext
!# <radiationLabel>
!# <label>CMB</label>
!# </radiationLabel>

!# <radiationSet>
!# <unitName>Radiation_Set_CMB</unitName>
!# <label>CMB</label>
!# </radiationSet>

!# <radiationTemperature>
!# <unitName>Radiation_Temperature_CMB</unitName>
!# <label>CMB</label>
!# </radiationTemperature>

!# <radiationFlux>
!# <unitName>Radiation_Flux_CMB</unitName>
!# <label>CMB</label>
!# </radiationFlux>
```

The first of these, \texttt{radiationLabel}, should contain a single element, \texttt{label}, which gives a label that will be used to identify this component, both in other directives and also in the internal parameters used to select this radiation component (e.g. in this case, a parameter \texttt{radiationTypeCMB} will be available within GALACTICUS to select the cosmic microwave background component). The other directives must all specify the same \texttt{label} element and additional give, in a \texttt{unitName} element, the name of a function/subroutine to be called to perform the relevant calculation.

The \texttt{radiationSet} directive must specify a subroutine with the following template:

```plaintext
subroutine Radiation_Set(componentMatched,thisNode,radiationProperties)
    implicit none
    logical, intent(in) :: componentMatched
    type(treeNode), intent(inout), pointer :: thisNode
    double precision, intent(inout), allocatable, dimension(:) :: radiationProperties
    if (.not.componentMatched) return
    .
    .
    return
```
If `componentMatched` is true, then the subroutine should set the radiation component, otherwise it should exit immediately. If the radiation component is to be set, then the routine can allocate the `radiationProperties` array as necessary to store any data needed to specify the radiation field. These data should then be set using, if necessary, any relevant information from `thisNode`.

The `radiationTemperature` directive should specify a subroutine with the following template:

```fortran
subroutine Radiation_Temperature(requestedType,ourType,radiationProperties,radiationTemperature,radiationType)
    implicit none
    integer, intent(in) :: requestedType,ourType
    double precision, intent(in), dimension(:) :: radiationProperties
    double precision, intent(inout) :: radiationTemperature
    integer, intent(in), optional, dimension(:) :: radiationType

    if (requestedType /= ourType) return
    if (present(radiationType)) then
        if (all(radiationType /= ourType)) return
    end if

    return
end subroutine Radiation_Temperature
```

The tests in the above should always be included so that the subroutine exits immediately if the component type is not active or not requested. Once these tests have been made, the subroutine should set the temperature (in units of Kelvin) of the radiation field (if applicable).

The `radiationFlux` directive should specify a subroutine with the following template:

```fortran
subroutine Radiation_Flux(requestedType,ourType,radiationProperties,wavelength,radiationFlux,radiationType)
    implicit none
    integer, intent(in) :: requestedType,ourType
    double precision, intent(in) :: wavelength
    double precision, intent(in), dimension(:) :: radiationProperties
    double precision, intent(inout) :: radiationFlux
    integer, intent(in), optional, dimension(:) :: radiationType

    if (requestedType /= ourType) return
    if (present(radiationType)) then
        if (all(radiationType /= ourType)) return
    end if

    return
end subroutine Radiation_Flux
```

The tests in the above should always be included so that the subroutine exits immediately if the component type is not active or not requested. Once these tests have been made, the subroutine should add the flux (in units of ergs cm$^2$ s$^{-1}$ Hz$^{-1}$ ster$^{-1}$) at the specified `wavelength` (in units of Å) of the radiation field to that in `radiationFlux`. 
Currently defined radiation component types are:

null A null component with no radiation.

CMB The cosmic microwave background, assumed to be a perfect blackbody spectrum with a temperature equal to \( T_{\text{CMB}}(1 + z) \).

IGB The intergalactic background light, set using the method selected by \( \text{radiationIntergalacticBackgroundMethod} \); see §17.4.1.

### Radiation Components: Intergalactic Background

Additional methods for the intergalactic background radiation component can be added using the \( \text{radiationIntergalacticBackgroundMethod} \) directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the file method is described by a directive:

```bash
!# <radiationIntergalacticBackgroundMethod>
!# <unitName>Radiation_IGB_File_Initialize</unitName>
!# </radiationIntergalacticBackgroundMethod>
```

Here, \( \text{Radiation_IGB_File_Initialize} \) is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```bash
subroutine Method_Initialize(radiationIntergalacticBackgroundMethod,Radiation_Set_Intergalactic_Background_Do,Radiation_Flux_Intergalactic_Background_Do)
implicit none
  type(varying_string), intent(in) :: radiationIntergalacticBackgroundMethod
  procedure(), pointer, intent(inout) :: Radiation_Set_Intergalactic_Background_Do,Radiation_Flux_Intergalactic_Background_Do

  if (radiationIntergalacticBackgroundMethod == 'myMethod') then
    Radiation_Set_Intergalactic_Background_Do => My_Method_Set
    Radiation_Flux_Intergalactic_Background_Do => My_Method_Flux
  end if
  return
end subroutine Method_Initialize
```

where \( \text{myMethod} \) is the name of this method as will be specified by the \( \text{radiationIntergalacticBackgroundMethod} \) input parameter. The procedure pointers \( \text{Radiation_Set_Intergalactic_Background_Do} \) and \( \text{Radiation_Flux_Intergalactic_Background_Do} \) must be set to point to subroutines which set the radiation field and return its flux as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The set subroutine must have the form:

```bash
subroutine My_Method_Set(thisNode,radiationProperties)
implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  double precision, intent(inout), allocatable, dimension(1:) :: radiationProperties

  return
end subroutine My_Method_Set
```

and should set the radiation component as described in §17.4.1. The flux subroutine must have the form:
subroutine My_Method_Flux(radiationProperties, wavelength, radiationFlux)
    implicit none
    double precision, intent(in) :: wavelength
    double precision, intent(in), dimension(:) :: radiationProperties
    double precision, intent(inout) :: radiationFlux

    return
end subroutine My_Method_Flux

and should increment radiationFlux as described in §17.4.1.

Currently defined intergalactic background radiation methods are:

file The intergalactic background radiation field, specified as a function of cosmic time, is read from a
file. The flux is determined by linearly interpolating to the required time and wavelength. The XML
file to read is specified by [radiationIGBFileName]. An example of the required file structure is:

    <spectrum>
      <URL>http://adsabs.harvard.edu/abs/1996ApJ...461...20H</URL>
      <description>Cosmic background radiation spectrum from quasars alone.</description>
      <source>Francesco Haardt on Aug 6 2005, via Cloudy 08.00</source>
      <wavelengths>
        <datum>0.0002481</datum>
        <datum>0.001489</datum>
        ...
        <units>Å</units>
      </wavelengths>
      <spectra>
        <datum>7.039E-49</datum>
        <datum>8.379E-48</datum>
        <datum>1.875E-39</datum>
        <datum>7.583E-38</datum>
        ...
        <redshift>0</redshift>
        <units>erg cm⁻² s⁻¹ Hz⁻¹ sr⁻¹</units>
      </spectra>
    </spectrum>

The optional URL, description, reference and source elements can be used to give the provenance of
the data. The wavelengths element should contain a set of datum elements each containing a wavelength
(in increasing order) at which the spectrum will be tabulated. Wavelengths must be given in Angstroms.
Multiple spectra elements can be given, each specifying the spectrum at a redshift as given in the
redshift element. Each spectra element must contain an array of datum elements that gives the
spectrum at each wavelength listed in the wavelength element. Spectra must be in units of erg cm⁻²
s⁻¹ Hz⁻¹ sr⁻¹.
17.4. Existing Method Types

Ram Pressure Mass Loss Rates in Disks/Spheroids

Additional methods for computing ram pressure induced mass loss rates in disks/spheroids can be added using the \texttt{ramPressureStrippingMassLossRate(Disks|Spheroids)}Method directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the \texttt{simple} method for disks is described by a directive:

\begin{verbatim}
!# <ramPressureStrippingMassLossRateDisksMethod>
!#    <unitName>Ram_Pressure_Stripping_Mass_Loss_Rate_Disks_Simple_Init</unitName>
!# </ramPressureStrippingMassLossRateDisksMethod>
\end{verbatim}

Here, \texttt{Ram_Pressure_Stripping_Mass_Loss_Rate_Disks_Simple_Init} is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

\begin{verbatim}
subroutine Method_Initialize(ramPressureStrippingMassLossRateDisksMethod,Ram_Pressure_Stripping_Mass_Loss_Rate_Disk_Get)
implicit none
type(varying_string), intent(in) :: starFormationTimescaleDisksMethod
procedure(), pointer, intent(inout) :: Ram_Pressure_Stripping_Mass_Loss_Rate_Disk_Get
if (ramPressureStrippingMassLossRateDisksMethod == 'myMethod') Ram_Pressure_Stripping_Mass_Loss_Rate_Disk_Get => My_Ram_Pressure_Stripping_Mass_Loss_Rate_Disk_Get
return
end subroutine Method_Initialize
\end{verbatim}

where \texttt{myMethod} is the name of this method as will be specified by the \texttt{ramPressureStrippingMassLossRate(Disks|Spheroids)} input parameter. The procedure pointer \texttt{Ram_Pressure_Stripping_Mass_Loss_Rate(Disks|Spheroids)} must be set to point to a function which returns mass loss rate due to ram pressure as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The mass loss rate function must have the form:

\begin{verbatim}
double precision function Ram_Pressure_Stripping_Mass_Loss_Rate_Disk_Get(thisNode)
implicit none
type(treeNode), intent(in) :: thisNode
. .
return
end function Ram_Pressure_Stripping_Mass_Loss_Rate_Disk_Get
\end{verbatim}

The function must return the mass loss rate induced by ram pressure forces (in units of $M_\odot$/Gyr) for the disk/spheroid in \texttt{thisNode}.

Currently defined ram pressure mass loss rate methods are:

\texttt{simple} The mass loss rate scales in proportion to the ratio of ram pressure and gravitational restoring forces;

\texttt{null} The mass loss rate is assumed to be always zero.

Satellite Merging Mass Movements

Additional methods for the satellite merging mass movements can be added using the \texttt{satelliteMergingMassMovementsMethod} directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the \texttt{simple} method is described by a directive:
17. Adding New Methods

!# <satelliteMergingMassMovementsMethod>
!# <unitName>Satellite_Merging_Mass_Movements_Simple.Initialize</unitName>
!# </satelliteMergingMassMovementsMethod>

Here, Satellite_Merging_Mass_Movements_Simple.Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

  implicit none
  type(varying_string), intent(in) :: satelliteMergingMassMovementsMethod
  procedure(), pointer, intent(inout) :: Satellite_Merging_Mass_Movement_Get

  if (satelliteMergingMassMovementsMethod == 'simple') Satellite_Merging_Mass_Movement_Get => My_Method_Get
  return
end subroutine Method.Initialize

where myMethod is the name of this method as will be specified by the satelliteMergingMassMovementsMethod input parameter. The procedure pointer Satellite_Merging_Mass_Movement_Get must be set to point to a function which sets the mass movement descriptors as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The mass movement subroutine must have the form:

subroutine My_Method_Get(thisNode,gasMovesTo,starsMoveTo,hostGasMovesTo,hostStarsMoveTo,mergerIsMajor)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  integer, intent(out) :: gasMovesTo,starsMoveTo,hostGasMovesTo,hostStarsMoveTo
  logical, intent(out) :: mergerIsMajor

 H H H
 return
end subroutine My_Method_Get

The subroutine must return values for each of the "MoveTo" descriptors to specify where stars and gas from thisNode and thisNode's host node should move to in the host. Allowed values are:

movesToDisk The material in question moves to the disk of the host node;
movesToSpheroid The material in question moves to the spheroid of the host node;
doesNotMove The material in question does not move (allowed only for host node descriptors).

Additionally, the mergerIsMajor flag should be set to indicate whether this merger is deemed to be "major" (typically defined as one which redistributes mass from a disk into a spheroidal component).

Currently defined satellite merger mass movement methods are:

verySimple In this case, the satellite is always added to the disk of the host, while material in the host does not move.
simple If the baryonic mass of the satellite exceeds a fraction majorMergerMassRatio of the baryonic mass of the host then all material is moved to the spheroid of the host. Otherwise, satellite gas moves to the component given by minorMergerGasMovesTo, satellite stars move to the host spheroid and host material does not move.
17.4. Existing Method Types

Baugh2005 If the baryonic mass of the satellite exceeds a fraction $\text{majorMergerMassRatio}$ of the baryonic mass of the host then all material is moved to the spheroid of the host. Otherwise, if the baryonic mass of the satellite exceeds a fraction $\text{burstMassRatio}$ of the baryonic mass of the host and the gas fraction in the host exceeds $\text{burstCriticalGasFraction}$ then all gas is moved to the host spheroid, while the host stellar disk remains in place. For mergers failing both criteria, satellite gas moves to the component given by $\text{minorMergerGasMovesTo}$, satellite stars move to the host spheroid and host material does not move.

Satellite Merging Remnant Sizes

Additional methods for the satellite merging remnant sizes can be added using the $\text{satelliteMergingRemnantSizeMethod}$ directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Cole2000 method is described by a directive:

```plaintext
!# <satelliteMergingRemnantSizeMethod>
!#   <unitName>Satellite_Merging_Remnant_Sizes_Cole2000_Initialize</unitName>
!# </satelliteMergingRemnantSizeMethod>
```

Here, $\text{Satellite_Merging_Remnant_Sizes_Cole2000_Initialize}$ is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
subroutine Method_Initialize(satelliteMergingRemnantSizeMethod,Satellite_Merging_Remnant_Size_Do)
  implicit none
  type(varying_string), intent(in) :: satelliteMergingRemnantSizeMethod
  procedure(), pointer, intent(inout) :: Satellite_Merging_Remnant_Size_Do

  if (satelliteMergingRemnantSizeMethod == 'myMethod') Satellite_Merging_Remnant_Size_Do => My_Method_Do
  return
end subroutine Method_Initialize
```

where $\text{myMethod}$ is the name of this method as will be specified by the $\text{satelliteMergingRemnantSizeMethod}$ input parameter. The procedure pointer $\text{Satellite_Merging_Remnant_Size_Do}$ must be set to point to a function which computes the size of the merger remnant and stores the properties (e.g. radius, circular velocity and specific angular momentum at the half-mass radius) of the host node. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The remnant size subroutine must have the form:

```plaintext
subroutine My_Method_Do(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode

  .
  .
  .
  return
end subroutine My_Method_Do
```

The subroutine must compute the properties of the merger remnant. Typically these are stored in the $\text{Satellite_Merging_Remnant_Sizes_Properties}$ module for later retrieval by the appropriate component.

Currently defined satellite merger remnant size methods are:
null  This is a null method which does nothing. It is useful for runs where no baryonic components are included (e.g. for studying dark matter only).

Cole2000  Implements the algorithm of Cole et al. [2000] to compute the remnant size. The orbital energy assumed can be adjusted using the mergerRemnantSizeOrbitalEnergy parameter, which is equivalent to the $f_{\text{orbit}}$ parameter of Cole et al. [2000].

Covington2008  Implements the algorithm of Covington et al. [2008] to compute the remnant size. The orbital energy assumed can be adjusted using the mergerRemnantSizeOrbitalEnergy parameter, which is equivalent to the $f_{\text{orbit}}$ parameter of Cole et al. [2000].

**Satellite Merging Remnants: Progenitor Properties**

Additional methods for satellite merging remnant progenitor properties can be added using the satelliteMergingRemnantProgenitorPropertiesMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the standard method is described by a directive:

```plaintext
!# <satelliteMergingRemnantProgenitorPropertiesMethod>
!# <unitName> Satellite_Merging_Remnant_Progenitor_Properties_Standard_Init</unitName>
!# </satelliteMergingRemnantProgenitorPropertiesMethod>
```

Here, Satellite_Merging_Remnant_Progenitor_Properties_Standard_Init is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
implicit none
    type(varying_string), intent(in) :: satelliteMergingRemnantProgenitorPropertiesMethod
    procedure(), pointer, intent(inout) :: Satellite_Merging_Remnant_Progenitor_Properties_Get


    return
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the satelliteMergingRemnantProgenitorPropertiesMethod input parameter. The procedure pointer Satellite_Merging_Remnant_Progenitor_Properties_Get must be set to point to a subroutine which computes various properties of the progenitor galaxies involved in the merger, as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The progenitor properties subroutine must have the form:

```plaintext
subroutine My_Method_Get(satelliteNode,hostNode,satelliteMass,hostMass,satelliteSpheroidMass &
    & ,hostSpheroidMass,hostSpheroidMassPreMerger,satelliteRadius,hostRadius &
    & ,angularMomentumFactor,remnantSpheroidMass,remnantSpheroidGasMass)
implicit none
    type(treeNode), intent(inout), pointer :: satelliteNode,hostNode
    double precision, intent(out) :: satelliteMass,hostMass,satelliteSpheroidMass, &
    & hostSpheroidMass,hostSpheroidMassPreMerger,satelliteRadius,hostRadius, &
    & angularMomentumFactor,remnantSpheroidMass,remnantSpheroidGasMass

    .
    .

    return
end subroutine My_Method_Do
```
17.4. Existing Method Types

The subroutine must compute properties of the merger progenitor galaxies in \texttt{satelliteNode} and \texttt{hostNode}: \texttt{satelliteMass} and \texttt{hostMass} are the total masses of the two galaxies; \texttt{satelliteSpheroidMass} and \texttt{hostSpheroidMass} are the masses of each galaxy that will end up in the spheroid of the merger remnant; \texttt{hostSpheroidMassPreMerger} is the mass of the host spheroid prior to the merger; \texttt{satelliteRadius} and \texttt{hostRadius} are radii of the two galaxies for use in merger remnant size calculations (and so should typically refer to the radius of material that will end up in the merger remnant spheroid); \texttt{remnantSpheroidMass} is the mass of the spheroid in the remnant; \texttt{remnantSpheroidGasMass} is the mass of gas in the spheroid of the remnant; and \texttt{angularMomentumFactor} gives the pseudo-specific angular momentum of the remnant in units of \((G \text{remnant.spheroid} \times \text{remnant.spheroid})^{1/2}\) where \(M_{\text{remnant.spheroid}}\) is the mass of the remnant spheroid and \(r_{\text{remnant.spheroid}}\) is the radius of the remnant spheroid.

Currently defined satellite merger progenitor properties methods are:

- **Cole2000** Implements the algorithm of Cole et al. [2000] to compute the remnant properties. Masses of host and spheroid are set equal to their stellar plus cold gas masses utilizing, while radii are the half-mass radii of each galaxy, including only those components which end up in the remnant spheroid. The angular momentum factor is set to a mass-weighted average of the corresponding factor for each component which will end up in the merger remnant spheroid.

- **standard** Masses of host and spheroid are set equal to their stellar plus cold gas masses utilizing, while radii are a mass-weighted average of the half-mass radii of the components which end up in the merger remnant spheroid. The angular momentum factor is similarly set to a mass-weighted average of the corresponding factor for each component which will end up in the merger remnant spheroid.

**Satellite Tidal Fields**

Additional methods for the satellite tidal fields can be added using the \texttt{satellitesTidalFieldMethod} directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the \texttt{sphericalSymmetry} method is described by a directive:

```fortran
!# <satellitesTidalFieldMethod>
!# <unitName></unitName>
!# </satellitesTidalFieldMethod>
```

Here, \texttt{Satellites_Tidal_Fields_Spherical_Symmetry_Initialize} is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(satellitesTidalFieldMethod,Satellites_Tidal_Field_Get)
  implicit none
  type (varying_string ) , intent(in ) :: satellitesTidalFieldMethod
  procedure(My_Method_Procedure), pointer, intent(inout) :: Satellites_Tidal_Field_Get

  if (satellitesTidalFieldMethod == 'myMethod') Satellites_Tidal_Field_Get => My_Method_Procedure
  return
end subroutine Method_Initialize
```

where \texttt{myMethod} is the name of this method as will be specified by the \texttt{satellitesTidalFieldMethod} input parameter. The procedure pointer \texttt{Satellites_Tidal_Field_Get} must be set to point to a function which returns the time until merging as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The satellite tidal field function must have the form:

```fortran
double precision function Satellites_Tidal_Fields(thisNode)
  implicit none
```

827
17. Adding New Methods

type(treeNode), pointer, intent(inout) :: thisNode

return
end function Satellites_Tidal_Fields

The function must return the magnitude of the tidal field for thisNode in units of (km/s)² Mpc⁻². Currently defined satellite tidal field methods are:

null Assumes a zero tidal field.
sphericalSymmetry Computes the tidal field assuming a spherically-symmetric host halo.

Satellite Virial Orbits

Additional methods for the satellite virial orbits (i.e. orbital parameters at virial radius crossing) can be added using the virialOrbitsMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Benson2005 method is described by a directive:

!# <virialOrbitsMethod>
!# <unitName>Virial_Orbital_Parameters_Benson2005_Initialize</unitName>
!# </virialOrbitsMethod>

Here, Virial_Orbital_Parameters_Benson2005_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

subroutine Method_Initialize(virialOrbitsMethod,Virial_Orbital_Parameters_Get)
  implicit none
  type(varying_string), intent(in) :: virialOrbitsMethod
  procedure(), pointer, intent(inout) :: Virial_Orbital_Parameters_Get

  if (virialOrbitsMethod.eq.'myMethod') Virial_Orbital_Parameters_Get => My_Method_Procedure
  return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the virialOrbitsMethod input parameter. The procedure pointer Virial_Orbital_Parameters_Get must be set to point to a subroutine which returns orbital parameters as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The orbital parameter subroutine must have the form:

function Virial_Orbital_Parameters(thisNode,hostNode,acceptUnboundOrbits) result (thisOrbit)
  implicit none
  type(keplerOrbit) :: thisOrbit
  type(treeNode), intent(inout), pointer :: thisNode,hostNode
  logical, intent(in) :: acceptUnboundOrbits

  return
end subroutine Virial_Orbital_Parameters
17.4. Existing Method Types

The subroutine must return a fully-defined Kepler orbit object (i.e. the orbit must have at least three parameters defined in addition to the node masses) initialized to the orbital parameters for thisNode orbitting in hostNode at the point of virial radius crossing. If acceptUnboundOrbits is true, then unbound orbits may be returned, otherwise, the routine must ensure that the returned orbit is bound. Velocities should be returned in units of km/s, lengthscales in units of Mpc and masses in $M_\odot$. Note that the usual conventions of the keplerOrbit object should be followed, namely the that orbitting bodies are treated as point masses, with the host being stationary and the usual reduced mass used.

Currently defined satellite virial orbit methods are:

- **Benson2005** The orbital parameters are select from the distribution found by Benson [2005].
- **Wetzel2010** The orbital parameters are select from the distribution found by Wetzel [2010].
- **fixed** The orbital parameters are set to fixed values, with $v_r = [\text{virialOrbitsFixedRadialVelocity}] V_{\text{virial}}$ and $v_\phi = [\text{virialOrbitsFixedTangentialVelocity}] V_{\text{virial}}$.

**Star Formation Feedback in Disks/Spheroids**

Additional methods for computing feedback from star formation in disks/spheroids can be added using the starFormationFeedback[Disks|Spheroids]Method directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the powerLaw method is described by a directive:

```
!# <starFormationFeedbackSpheroidsMethod>
!# <unitName>Star_Formation_Feedback_Spheroids_Power_Law.Initialize</unitName>
!# </starFormationFeedbackSpheroidsMethod>
```

Here, Star_Formation_Feedback_Spheroids_Power_Law.Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(starFormationFeedbackDisksMethod, Star_Formation_Feedback_Disk_Outflow_Rate_Get)
implicit none
type(varying_string), intent(in) :: starFormationFeedbackDisksMethod
procedure(), pointer, intent(inout) :: Star_Formation_Feedback_Disk_Outflow_Rate_Get

if (starFormationFeedbackDisksMethod == 'myMethod') Star_Formation_Feedback_Disk_Outflow_Rate_Get => My_Star_Formation_Feedback_Disk_Outflow_Rate_Get
return
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the starFormationFeedback[Disks|Spheroids]Method input parameter. The procedure pointer Star_Formation_Feedback_Disk_Outflow_Rate_Get (or Star_Formation_Feedback_Spheroid_Outflow_Rate_Get for the spheroid case) must be set to point to a function which returns the mass outflow rate due to star formation as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The outflow rate function must have the form:

```fortran
double precision function Star_Formation_Feedback_Outflow_Rate_Get(thisNode, starFormationRate, energyInputRate)
implicit none
type(treeNode), intent(inout), pointer :: thisNode
double precision, intent(in) :: starFormationRate, energyInputRate
```

...
17. Adding New Methods

```
return
end function Star_Formation_Feedback_Outflow_Rate_Get
```

The function must return the mass outflow rate (in \( M_\odot \text{ Gyr}^{-1} \)) for `thisNode`.

Currently defined star formation feedback methods are:

**fixed** The outflow rate is a fixed multiple of the the star formation rate.

**powerLaw** The outflow rate is given by

\[
\dot{M}_{\text{outflow}} = \left( \frac{V_{\text{outflow}}}{V} \right)^{\alpha_{\text{outflow}}} \frac{\dot{E}}{E_{\text{canonical}}},
\]

(17.8)

where \( V_{\text{outflow}} = [\text{disk|spheroid}]\text{OutflowVelocity} \) (in km/s) and \( \alpha_{\text{outflow}} = [\text{disk|spheroid}]\text{OutflowVelocity} \) are input parameters, \( V \) is the characteristic velocity of the component, \( \dot{E} \) is the rate of energy input from stellar populations and \( E_{\text{canonical}} \) is the total energy input by a canonical stellar population normalized to \( 1M_\odot \) after infinite time.

**Creasey2012** The outflow rate computed using the model of Creasey et al. [2012].

**(Expulsive) Star Formation Feedback in Disks/Spheroids**

Additional methods for computing expulsive feedback\(^8\) from star formation in disks/spheroids can be added using the `starFormationExpulsiveFeedback[Disks|Spheroids]Method` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method.

For example, the `powerLaw` method is described by a directive:

```
!# <starFormationExpulsiveFeedbackSpheroidsMethod>
!#  <unitName>Star_Formation_Expulsive_Feedback_Spheroids_Power_Law_Initialize</unitName>
!# </starFormationExpulsiveFeedbackSpheroidsMethod>
```

Here, `Star_Formation_Expulsive_Feedback_Spheroids_Power_Law_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```
subroutine Method_Initialize(starFormationExpulsiveFeedbackDisksMethod, Star_Formation_Expulsive_Feedback_Disk_Outflow_Rate_Get)
implicit none
type(varying_string), intent(in) :: starFormationFeedbackDisksMethod
procedure(), pointer, intent(inout) :: Star_Formation_Expulsive_Feedback_Disk_Outflow_Rate_Get

if (starFormationExpulsiveFeedbackDisksMethod == 'myMethod') Star_Formation_Expulsive_Feedback_Disk_Outflow_Rate_Get => My_Star_Formation_Expulsive_Feedback_Disk_Outflow_Rate_Get
return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `starFormationExpulsiveFeedback[Disks|Spheroids]` input parameter. The procedure pointer `Star_Formation_Expulsive_Feedback_Disk_Outflow_Rate_Get` (or `Star_Formation_Expulsive_Feedback_Spheroid_Outflow_Rate_Get` for the spheroid case) must be set to point to a function which returns the expulsive mass outflow rate due to star formation as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The outflow rate function must have the form:

\(^8\)“Expulsive” feedback implies outflows in which gas is driven not only out of a galaxy but also out of its host dark matter halo.
double precision function Star_Formation_Expulsive_Feedback_Outflow_Rate_Get(thisNode, starFormationRate, energyInputRate)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  double precision, intent(in) :: starFormationRate, energyInputRate
  .
  .
  return
end function Star_Formation_Expulsive_Feedback_Outflow_Rate_Get

The function must return the expulsive mass outflow rate (in $M_\odot$ Gyr$^{-1}$) for thisNode.

Currently defined star formation expulsive feedback methods are:

null Assumes a zero outflow rate.

superwind The outflow rate is given by

$$\dot{M}_{\text{outflow}} = \beta_{\text{superwind}} \frac{\dot{E}}{E_{\text{canonical}}} \begin{cases} \left(\frac{V_{\text{superwind}}}{V}\right)^2 & \text{if } V > V_{\text{superwind}} \\ 1 & \text{otherwise,} \end{cases}$$ \hfill (17.9)

where $V_{\text{superwind}} = \left[\text{disk|spheroid}\right]$ SuperwindVelocity (in km/s) and $\beta_{\text{superwind}} = \left[\text{disk|spheroid}\right]$ SuperwindMassLoading are input parameters, $V$ is the characteristic velocity of the component, $\dot{E}$ is the rate of energy input from stellar populations and $E_{\text{canonical}}$ is the total energy input by a canonical stellar population normalized to $1M_\odot$ after infinite time.

Star Formation Rate Surface Density in Disks

Additional methods for computing the surface density of star formation rate in disks can be added using the starFormationRateSurfaceDensityDisksMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Blitz-Rosolowsky2006 method is described by a directive:

```
!# <starFormationRateSurfaceDensityDisksMethod>
!# <unitName>Star_Formation_Rate_Surface_Density_Disks_BR_Initialize</unitName>
!# </starFormationRateSurfaceDensityDisksMethod>
```

Here, Star_Formation_Rate_Surface_Density_DISKS_BR_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```subroutine Method_Initialize(starFormationRateSurfaceDensityDisksMethod, Star_Formation_Rate_Surface_Density_Disk_Get)
  implicit none
  type(varying_string), intent(in) :: starFormationRateSurfaceDensityDisksMethod
  procedure(), pointer, intent(inout) :: Star_Formation_Rate_Surface_Density_Disk_Get

  if (starFormationRateSurfaceDensityDisksMethod == 'myMethod') Star_Formation_Rate_Surface_Density_Disk_Get => My_Method_Get

  return
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the starFormationRateSurfaceDensityDisksMethod input parameter. The procedure pointer Star_Formation_Rate_Surface_Density_Disk_Get must be set to point to a function which returns the surface density of star formation rate at a specified radius as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The star formation rate surface density function must have the form:
double precision function Star_Formation_Rate_Surface_Density_Get(thisNode, radius)
  implicit none
  type (treeNode), intent(inout), pointer :: thisNode
  double precision, intent(in) :: radius
  return
end function Star_Formation_Rate_Surface_Density_Get

The function must return the surface density of star formation rate (in units of $M_\odot$ Gyr$^{-1}$ Mpc$^{-2}$) for thisNode.

Currently defined star formation rate surface density methods are:

**Kennicutt-Schmidt** The rate is given by the Kennicutt-Schmidt law (Schmidt 1959, Kennicutt 1998; see §13.38.1).

**extendedSchmidt** The rate is given by the extended Schmidt law (Shi et al. 2011; see §13.38.2).

**Blitz-Rosolowsky2006** The rate is given by the Blitz-Rosolowsky rule (Blitz and Rosolowsky 2006; see §13.38.3).

**KMT09** The rate is given by the Krumholz-McKee-Tumlinson (Krumholz et al. 2009; see §13.38.4);

### Star Formation Timescale in Disks/Spheroids

Additional methods for computing star formation timescales in disks/spheroids can be added using the `starFormationTimescale[Disks|Spheroids]Method` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `dynamicalTime` method is described by a directive:

```
!# <starFormationTimescaleDisksMethod>
!# <unitName>Star_Formation_Timescale_Disks_Dynamical_Time_Initialize</unitName>
!# </starFormationTimescaleDisksMethod>
```

Here, `Star_Formation_Timescale_Disks_Dynamical_Time_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```
subroutine Method_Initialize(starFormationTimescaleDisksMethod, Star_Formation_Timescale_Disk_Get)
  implicit none
  type(varying_string), intent(in) :: starFormationTimescaleDisksMethod
  procedure(), pointer, intent(inout) :: Star_Formation_Timescale_Disk_Get

  if (starFormationTimescaleDisksMethod == 'myMethod') Star_Formation_Timescale_Disk_Get => My_Method_Get_Procedure
return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `starFormationTimescale[Disks|Spheroids]Method` input parameter. The procedure pointer `Star_Formation_Timescale_Disk_Get` (or `Star_Formation_Timescale_Spheroid_Get` for the spheroid case) must be set to point to a function which returns the timescale for star formation as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The star formation timescale function must have the form:
double precision function Star_Formation_Timescale_Get(thisNode)
  implicit none
  type(treeNode), intent(in) :: thisNode
  .
  .
  .
  return
end function Star_Formation_Timescale_Get

The function must return the star formation timescale (in units of Gyr) for thisNode.

Currently defined star formation timescale methods are:

haloScaling  The timescale scales with halo virial velocity and redshift;

fixed  The timescale is a fixed quantity;

dynamicalTime  The timescale is given by

\[ \tau_\star = \epsilon_\star^{-1} \tau_{\text{dynamical}, \text{disk|spheroid}} \left( \frac{V_{\text{disk|spheroid}}}{200 \text{ km/s}} \right)^{\alpha_\star}, \]  

(17.10)

where \( \epsilon_\star = \text{starFormation[Disk|Spheroid]Efficiency} \) is a star formation efficiency and \( \alpha_\star = \text{starFormation[Disk|Spheroid]VelocityExponent} \) controls the scaling with velocity. Note that \( \tau_{\text{dynamical}, \text{disk|spheroid}} = R_{\text{disk|spheroid}} / V_{\text{disk|spheroid}} \) where the radius and velocity are whatever characteristic values returned by the disk/spheroid method. This scaling is functionally similar to that adopted by Cole et al. [2000], but they specifically used the half-mass radius and circular velocity at that radius.

Baugh2005]  The timescale is given by \( \tau_0 (V_{\text{disk}} / 200 \text{ km/s})^{\alpha_a \beta} \).

integratedSurfaceDensity]  The timescale is given by \( \tau_\star = M_{\text{cold}} / \int_0^\infty 2\pi r \dot{\Sigma}_\star (r) dr \) where \( \dot{\Sigma}_\star (r) \) is the surface density of star formation rate (see §13.38)

Stellar Astrophysics

Additional methods for stellar astrophysical properties can be added using the stellarAstrophysicsMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the file method is described by a directive:

!# <stellarAstrophysicsMethod>
!#   <unitName>Stellar_Astrophysics_File_Initialize</unitName>
!# </stellarAstrophysicsMethod>

Here, Stellar_Astrophysics_File_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

  implicit none
  type(varying_string), intent(in) :: stellarAstrophysicsMethod

  if (stellarAstrophysicsMethod == 'myMethod') then
    Star_Ejected_Mass_Get => My_Star_Ejected_Mass
    Star_Initial_Mass_Get => My_Star_Initial_Mass
    Star_Metal_Yield_Mass_Get => My_Star_Metal_Yield_Mass
    Star_Lifetime_Get => My_Star_Lifetime
  else
    ! # myMethod subroutine
  end if
end subroutine Method_Initialize

833
17. Adding New Methods

    Star_Lifetime_Get => My_Star_Lifetime
end if
return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the stellarAstrophysicsMethod input parameter. The procedure pointers must be set to point to functions which return stellar properties as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The ejected mass and lifetime functions must have the form:

    double precision function Star_Property(initialMass,metallicity)
        implicit none
        double precision, intent(in) :: initialMass,metallicity
        .
        .
        .
        return
    end function Star_Property

These functions must return the total ejected mass (in $M_\odot$), total metal yield (in $M_\odot$) and lifetime (in Gyr) for a star of the specified initialMass and metallicity.

The metal yield function must have the form:

    double precision function Star_Metal_Yield(initialMass,metallicity,atomIndex)
        implicit none
        double precision, intent(in) :: initialMass,metallicity
        integer, intent(in), optional :: atomIndex
        .
        .
        .
        return
    end function Star_Property

This function must return the yield (in $M_\odot$) of the element identified by atomIndex (as returned by the Atom_Lookup() function from the Atomic_Data module) if present, or total metal yield otherwise for a star of the specified initialMass and metallicity.

The initial mass function must have the form:

    double precision function Star_Initial_Mass(lifetime,metallicity)
        implicit none
        double precision, intent(in) :: lifetime,metallicity
        .
        .
        .
        return
    end function Star_Initial_Mass

and should return the initial mass (in $M_\odot$) of a star of given lifetime (specified in Gyr) and metallicity.

Currently defined stellar astrophysics methods are:

file Stellar properties are read from an XML file and interpolated. The structure of the XML file is described in §13.43.1.
Stellar Population Properties

Additional methods for computing properties of stellar populations can be added using the `stellarPopulationPropertiesMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `instantaneous` method is described by a directive:

```plaintext
!# <stellarPopulationPropertiesMethod>
!# <unitName>Stellar_Population_Properties_Instantaneous_Initialize</unitName>
!# </stellarPopulationPropertiesMethod>
```

Here, `Stellar_Population_Properties_Instantaneous_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
  implicit none
  type(varying_string), intent(in) :: stellarPopulationPropertiesMethod
  if (stellarPopulationPropertiesMethod == 'myMethod') then
    Stellar_Population_Properties_History_Create_Do => My_Method_History_Create_Procedure
  end if
  return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `stellarPopulationPropertiesMethod` input parameter. The procedure pointers `Stellar_Population_Properties_Rates_Get` and `Stellar_Population_Properties_Scales_Get` must be set to point to subroutines which return properties of a stellar population and set scaling factors for ODE error control as described below, while the `Stellar_Population_Properties_History_Count_Get` and `Stellar_Population_Properties_History_Create_Do` procedure pointers must be set to point to functions which return the number of histories that will be required by this method and create a suitable history object respectively. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The stellar populations properties subroutine must have the form:

```plaintext
subroutine Stellar_Population_Properties_Rates(starFormationRate,fuelAbundances,component,thisNode,thisHistory,stellarMassRate&,stellarAbundancesRates,stellarLuminositiesRates,fuelMassRate,fuelAbundancesRates,energyInputRate)
  implicit none
  implicit none
  double precision, intent(out) :: stellarMassRate,fuelMassRate,energyInputRate
  type(abundancesStructure), intent(out) :: stellarAbundancesRates,fuelAbundancesRates
  integer, intent(in) :: component
  double precision, dimension(:) :: stellarLuminositiesRates
  double precision, intent(in) :: starFormationRate
  type(abundancesStructure), intent(in) :: fuelAbundances
  type(treeNode), intent(inout), pointer :: thisNode
```
17. Adding New Methods

```plaintext
type(history), intent(inout) :: thisHistory
.
.
.
return
end subroutine Stellar_Population_Properties_Rates
```

The subroutine is given the starFormationRate (in $M_\odot$ Gyr$^{-1}$) in thisNode. Any history information required by this method must be passed in via the history argument. Stars are forming from fuel material with composition specified by fuelAbundances and occurring in the specified galactic component (using the labels provided by the Galactic_Structure_Options module). The subroutine must return the rates of change of stellar and fuel mass (in $M_\odot$ Gyr$^{-1}$) in stellarMassRate and fuelMassRate respectively, and the corresponding rates (also in $M_\odot$ Gyr$^{-1}$) of abundance change in stellarAbundancesRates and fuelAbundancesRates respectively. Finally, it should return rates of change (in $L_{AB}$ Gyr$^{-1}$) of stellar luminosities for all requested output bands in stellarLuminositiesRates. Additionally, the rate of energy input from stellar populations must be returned in energyInputRate.

The scales procedure should have the form:

```plaintext
subroutine Stellar_Population_Properties_Scales_Noninstantaneous(thisHistory,stellarMass,stellarAbundances)
    implicit none
double precision, intent(in) :: stellarMass
type(abundancesStructure), intent(in) :: stellarAbundances
type(history), intent(inout) :: thisHistory
.
.
return
end subroutine Stellar_Population_Properties_Scales_Noninstantaneous
```

and should set scale factors for ODE error control (see §17.3.3) in the stellar population properties history thisHistory. The stellarMass and stellarAbundances (both in $M_\odot$) are provided as input as they are often useful in choosing appropriate scale factors.

The history count function must have the form

```plaintext
integer function Stellar_Population_Properties_History_Count()
    implicit none
.
.
return
end function Stellar_Population_Properties_History_Count
```

and should return the number of histories that will be required by this method. The history create function must have the form

```plaintext
subroutine Stellar_Population_Properties_History_Create(thisNode,thisHistory)
type(treeNode), intent(inout), pointer :: thisNode
type(history), intent(inout) :: thisHistory
.
.
return
end subroutine Stellar_Population_Properties_History_Create
```
and should create thisHistory with a suitable set of time steps for thisNode.

Currently defined stellar population properties methods are:

**instantaneous** Computes stellar population properties using an instantaneous recyclying approxima-

**Stellar Population Spectra**

Additional methods for computing spectra of stellar populations can be added using the `stellarPopulationSpectraMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Conroy-White-Gunn2009 method is described by a directive:

```plaintext
!# <stellarPopulationSpectraMethod>
!# <unitName>Stellar_Population_Spectra_Conroy_Initialize</unitName>
!# </stellarPopulationSpectraMethod>
```

Here, `Stellar_Population_Spectra_Conroy_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
implicit none
  type(varying_string), intent(in) :: stellarPopulationSpectraMethod
  procedure(), pointer, intent(inout) :: Stellar_Population_Spectra_Get,Stellar_Population_Spectrum_Tabulation_Get
  if (stellarPopulationSpectraMethod == 'myMethod') then
    Stellar_Population_Spectra_Get => My_Method_Spectra_Get
  end if
return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `stellarPopulationSpectraMethod` input parameter. The procedure pointer `Stellar_Population_Spectra_Get` must be set to point to a function which returns the spectrum of a stellar population as described below while the `Stellar_Population_Spectrum_Tabulation_Get` pointer must be set to point to a subroutine which returns a tabulation of ages and metallicities on which stellar spectra should be tabulated. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The stellar spectra function must have the form:

```plaintext
double precision function Stellar_Population_Spectra_Get(abundances,age,wavelength,imfIndex)
imPLICIT none
  type(abundancesStructure), intent(in) :: abundances
  double precision, intent(in) :: age,wavelength
  integer, intent(in) :: imfIndex
  .
  .
return
end function Stellar_Population_Spectra_Get
```

The function is given the `abundances`, `age` (in Gyr), and `imfIndex` of the stellar population and the `wavelength` (in Å) at which the spectrum should be computed. The spectrum should be returned in units of $L_\odot$ Hz$^{-1}$.
The tabulation subroutine must have the form:

```fortran
subroutine Stellar_Population_Spectrum_Tabulation(imfIndex, agesCount, metallicitiesCount, age, metallicity)
  implicit none
  integer, intent(in) :: imfIndex
  integer, intent(out) :: agesCount, metallicitiesCount
  double precision, intent(out), allocatable, dimension(:) :: age, metallicity
  
  return
end subroutine Stellar_Population_Spectrum_Tabulation
```

and should return the number of ages and metallicities at which stellar population spectra should be tabulated for the specified IMF, and should allocate the `age` and `metallicity` arrays appropriately and should fill them with the ages and metallicities at which to tabulate.

Currently defined stellar population properties methods are:

- **Conroy-White-Gunn2009**

  Uses the FSPS code of Conroy et al. [2009] to compute stellar spectra. If necessary, the FSPS code will be downloaded, patched and compiled and run to generate spectra. These tabulations are then stored to file for later retrieval.

- **File**

  Stellar spectra for a given IMF are read from the file specified by the `stellarPopulationSpectraForXXXXIMF` where `XXXX` is the name of the IMF. This should specify an HDF5 file with the following structure:

  - `ages` Dataset `{ageCount}`
  - `metallicities` Dataset `{metallicityCount}`
  - `spectra` Dataset `{metallicityCount, ageCount, metallicityCount}`
  - `wavelengths` Dataset `{wavelengthCount}`

  where the datasets contain the tabulated ages (in Gyr), metallicities (logarithmic, relative to Solar), wavelengths (in Å) and spectra (in $L_\odot \text{Hz}^{-1}$).

Currently, the following pre-computed stellar spectra files are available as a separate download from [http://users.obs.carnegiescience.edu/abenson/galacticus/data/Galacticus_SSP_Data.tar.bz2](http://users.obs.carnegiescience.edu/abenson/galacticus/data/Galacticus_SSP_Data.tar.bz2):

- data/stellarPopulations/SSP_Spectra_Conroy-et-al_v2.0_imfSalpeter.hdf5 corresponds to a Salpeter IMF computed using v2.0 of the FSPS code;
- data/stellarPopulations/SSP_Spectra_Conroy-et-al_v2.1_imfSalpeter.hdf5 corresponds to a Salpeter IMF computed using v2.1 of the FSPS code;
- data/stellarPopulations/SSP_Spectra_Conroy-et-al_v2.1_imfChabrier.hdf5 corresponds to a Chabrier IMF computed using v2.1 of the FSPS code;
- data/stellarPopulations/SSP_Spectra_Conroy-et-al_v2.2_imfChabrier.hdf5 corresponds to a Chabrier IMF computed using v2.2 of the FSPS code;
- data/stellarPopulations/SSP_Spectra_Conroy-et-al_v2.2_imfKennicutt.hdf5 corresponds to a Kennicutt IMF computed using v2.2 of the FSPS code;
- data/stellarPopulations/SSP_Spectra_Conroy-et-al_v2.2_imfBaugh2005TopHeavy.hdf5 corresponds to the top-heavy IMF of Baugh et al. [2005] computed using v2.2 of the FSPS code;
17.4. Existing Method Types

data/stellarPopulations/SSP_Spectra_Maraston_hbMorphologyRed_imfKroupa.hdf5  The spectra from Maraston [2005] for a Kroupa IMF and a red horizontal branch morphology;

data/stellarPopulations/SSP_Spectra_Maraston_hbMorphologyRed_imfSalpeter.hdf5  The spectra from Maraston [2005] for a Salpeter IMF and a red horizontal branch morphology;


data/stellarPopulations/SSP_Spectra_BC2003_highResolution_imfSalpeter.hdf5  The (high resolution) spectra from Bruzual and Charlot [2003] for a Salpeter IMF, using Padova 1994 tracks;


data/stellarPopulations/SSP_Spectra_Grasil_gkn15rd_ken.hdf5  The spectra used by GRASIL for a Kennicutt IMF;

data/stellarPopulations/SSP_Spectra_Grasil_gkn1rd_ken.hdf5  The spectra used by GRASIL for a Kennicutt IMF;

data/stellarPopulations/SSP_Spectra_Grasil_gsrdk0b_sal.hdf5  The spectra used by GRASIL for a Salpeter IMF;

data/stellarPopulations/SSP_Spectra_Grasil_imf27_kro.hdf5  Spectra used by GRASIL.

Note that the high resolution spectra from Bruzual and Charlot [2003] may require you to adjust the [stellarPopulationLuminosityIntegrationToleranceRelative] parameter to a larger value. The sharp features in these high resolution spectra can be difficult to integrate. Scripts to convert the data provided by Maraston [2005] and Bruzual and Charlot [2003] into GALACTICUS’s format are provided in the scripts/ssps folder. Spectra for other initial mass functions will be computed automatically when using the Conroy et al. [2009] population synthesis models.

Stellar Feedback

Additional methods for stellar feedback can be added using the stellarFeedbackMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the standard method is described by a directive:

    !# <stellarFeedbackMethod>
    !# <unitName>Stellar_Feedback_Standard_Initialize</unitName>
    !# </stellarFeedbackMethod>

Here, Stellar_Feedback_Standard_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

    subroutine Method_Initialize(stellarFeedbackMethod,Stellar_Feedback_Cumulative_Energy_Input_Get)
    implicit none
    type(varying_string), intent(in) :: stellarFeedbackMethod
    procedure(), pointer, intent(inout) :: Stellar_Feedback_Cumulative_Energy_Input_Get

    if (stellarFeedbackMethod == 'myMethod') then

839
17. Adding New Methods


. .
end if
return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the stellarFeedbackMethod input parameter. The procedure pointer Stellar_Feedback_Cumulative_Energy_Input_Get must be set to point to a function which returns the cumulative energy input from stars as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The function must have the form:

```plaintext
double precision function Stellar_Feedback_Cumulative_Energy_Input(initialMass,age,metallicity)
  implicit none
  double precision, intent(in) :: initialMass,age,metallicity
  . .
  return
end function Stellar_Feedback_Cumulative_Energy_Input
```

The function must return the cumulative energy input (in \(M_\odot (\text{km/s})^2\)) from stars of given initialMass and metallicity after a time age. Currently defined stellar feedback methods are:

**standard** This method assumes that the energy input has contributions from stellar winds, Type Ia, Type II and Population III supernovae. The minimum mass required for a star to produce a Type II supernova is specified via initialMassForSupernovaeTypeII (in \(M_\odot\)), while the energy per Type II or Ia supernova is specified via supernovaEnergy (in ergs).

**Stellar Tracks**

Additional methods for stellar tracks can be added using the stellarTracksMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the file method is described by a directive:

```plaintext
!# <stellarTracksMethod>
!# <unitName>Stellar_Tracks_Initialize_File</unitName>
!# </stellarTracksMethod>
```

Here, Stellar_Tracks_Initialize_File is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```plaintext
subroutine Method_Initialize(stellarTracksMethod,Stellar_Luminosity_Get,Stellar_Effective_Temperature_Get)
  implicit none
  type(varying_string), intent(in) :: stellarTracksMethod
  procedure(), pointer, intent(inout) :: Stellar_Luminosity_Get,Stellar_Effective_Temperature_Get

  if (stellarTracksMethod == 'myMethod') then
    Stellar_Luminosity_Get => My_Stellar_Luminosity_Get
  .
end subroutine Method_Initialize
```
17.4. Existing Method Types

Stellar_Effective_Temperature_Get => My_Stellar_Effective_Temperature_Get

end if
return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the stellarTracksMethod input parameter. The procedure pointers Stellar_Luminosity_Get and Stellar_Effective_Temperature_Get must be set to point to functions which return the luminosity and effective temperatures of stars as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The functions must have the form:

\[
\text{double precision function Stellar_Tracks_Function(initialMass,age,metallicity) }
\]

\[
\text{implicit none}
\]

\[
\text{double precision, intent(in)} :: \text{initialMass,age,metallicity}
\]

return
end function Stellar_Tracks_Function

The luminosity function must return the bolometric luminosity (in \(L_\odot\)) of a star of given initialMass and metallicity after a time age. The effective temperature function should give the effective temperature (in Kelvin) for the same star.

Currently defined stellar tracks methods are:

**file** Stellar tracks are read from an HDF5 file and interpolated in. The structure of the HDF5 is described in §13.43.3.

**Stellar Winds**

Additional methods for stellar winds can be added using the stellarWindsMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Leitherer1992 method is described by a directive:

\[
!\# <\text{stellarWindsMethod}>
!\# <\text{unitName}>Stellar_Winds_Leitherer1992_Initialize</\text{unitName}>
!\# </\text{stellarWindsMethod}>
\]

Here, Stellar_Winds_Leitherer1992_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

subroutine Method_Initialize(stellarWindsMethod,Stellar_Winds_Mass_Loss_Rate_Get,Stellar_Winds_Terminal_Velocity_Get)
implicit none
type(varying_string), intent(in) :: stellarWindsMethod
procedure(), pointer, intent(inout) :: Stellar_Winds_Mass_Loss_Rate_Get,Stellar_Winds_Terminal_Velocity_Get
if (stellarWindsMethod == 'myMethod') then
    Stellar_Winds_Mass_Loss_Rate_Get => My_Stellar_Winds_Mass_Loss_Rate_Get
    Stellar_Winds_Terminal_Velocity_Get => My_Stellar_Winds_Terminal_Velocity_Get
end if
return
end subroutine Method_Initialize
17. Adding New Methods

. .

end if
return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the stellarWindsMethod input parameter. The procedure pointers Stellar_Winds_Mass_Loss_Rate_Get and Stellar_Winds_Terminal_Velocity_Get must be set to point to functions which return the mass loss rate and terminal velocity of winds as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The functions must have the form:

double precision function Stellar_Wind_Function(initialMass, age, metallicity)
   implicit none
   double precision, intent(in) :: initialMass, age, metallicity
   .
   return
end function Stellar_Wind_Function

The mass loss function must return the rate of mass loss (in $M_\odot$/Gyr) from stars of given initialMass and metallicity after a time age. The terminal velocity function should give the velocity (in km/s) at infinity of the wind for the same stars.

Currently defined stellar winds methods are:

Leitherer1992 Computes wind properties using the fitting functions of Leitherer et al. [1992] and GALACTICUS stellar tracks.

Supernovae Type Ia

Additional methods for Type Ia supernovae can be added using the supernovaeIaMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Nagashima method is described by a directive:

!# <supernovaeIaMethod>
!# <unitName>Supernovae_Type_Ia_Nagashima_Initialize</unitName>
!# </supernovaeIaMethod>

Here, Supernovae_Type_Ia_Nagashima_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

subroutine Method_Initialize(supernovaeIaMethod, SNeIa_Cumulative_Number_Get, SNeIa_Cumulative_Yield_Get)
   implicit none
   type(varying_string), intent(in) :: supernovaeIaMethod
   procedure(), pointer, intent(inout) :: SNeIa_Cumulative_Number_Get, SNeIa_Cumulative_Yield_Get
   .
   if (supernovaeIaMethod == 'myMethod') then
      SNeIa_Cumulative_Number_Get => My_SNeIa_Cumulative_Number_Get
      SNeIa_Cumulative_Yield_Get  => My_SNeIa_Cumulative_Yield_Get
   .
where `myMethod` is the name of this method as will be specified by the `supernovaeIaMethod` input parameter. The procedure pointers `SNeIa_Cumulative_Number_Get` and `SNeIa_Cumulative_Yield_Get` must be set to point to functions which return the cumulative number of and cumulative yield from Type Ia supernovae as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The cumulative number function must have the form:

```fortran
double precision function SNeIa_Cumulative_Number(initialMass,age,metallicity)
  implicit none
  double precision, intent(in) :: initialMass,age,metallicity
  .
  .
  .
  return
end function SNeIa_Cumulative_Number
```

and must return the number of Type Ia supernovae resulting per $M_\odot$ of stars formed with given `initialMass` and `metallicity` after a time `age`. (Since Type Ia's form in binary systems this function should specifically return the number such that when integrated over the IMF it gives the correct total number of Type Ia supernovae formed from a single stellar population.)

The cumulative yield function must have the form:

```fortran
double precision function SNeIa_Cumulative_Yield(initialMass,age,metallicity,atomIndex)
  implicit none
  double precision, intent(in) :: initialMass,age,metallicity
  integer, intent(in), optional :: atomIndex
  .
  .
  .
  return
end function SNeIa_Cumulative_Yield
```

and should return the yield of the element identified by `atomIndex` (as returned by the `Atom.Lookup()` function from the `Atomic_Data` module) if present, or total metal yield otherwise from Type Ia's resulting from stars defined in the same way as for the cumulative number function.

Currently defined type Ia supernovae methods are:

**Nagashima** Computes Type Ia properties using the methods described by Nagashima et al. [2005].

**Tidal Mass Loss Rates in Disks/Spheroids**

Additional methods for computing tidal induced mass loss rates in disks/spheroids can be added using the `tidalStrippingMassLossRate(Disks|Spheroids)Method` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `simple` method for disks is described by a directive:
Here, Tidal_Stripping_Mass_Loss_Rate_Disks_Simple_Init is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(tidalStrippingMassLossRateDisksMethod,Tidal_Stripping_Mass_Loss_Rate_Disk_Get)
  implicit none
  type(varying_string), intent(in) :: starFormationTimescaleDisksMethod
  procedure(), pointer, intent(inout) :: Tidal_Stripping_Mass_Loss_Rate_Disk_Get

  if (tidalStrippingMassLossRateDisksMethod == 'myMethod') Tidal_Stripping_Mass_Loss_Rate_Disk_Get => My_Tidal_Stripping_Mass_Loss_Rate_Disk_Get
  return
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the tidalStrippingMassLossRate(Disks|Spheroids)Method input parameter. The procedure pointer Tidal_Stripping_Mass_Loss_Rate_(Disk|Spheroid)_Get must be set to point to a function which returns mass loss rate due to tidal forces as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The mass loss rate function must have the form:

```fortran
double precision function Tidal_Stripping_Mass_Loss_Rate_Disk_Get(thisNode)
  implicit none
  type(treeNode), intent(in) :: thisNode
  
  return
end function Tidal_Stripping_Mass_Loss_Rate_Disk_Get
```

The function must return the mass loss rate induced by tidal forces (in units of $M_\odot$/Gyr) for the disk/spheroid in thisNode.

Currently defined ram pressure mass loss rate methods are:

- **simple** The mass loss rate scales in proportion to the ratio of tidal and gravitational restoring forces;
- **null** The mass loss rate is assumed to be always zero.

### Survey Geometry

Additional methods for survey geometries can be added using the `surveyGeometryMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Martin-2010-ALFALFA method is described by a directive:

```fortran
!# <surveyGeometryMethod>
!# <unitName>Geometry_Surveys_MARTIN_2010_ALFALFA_Initialize</unitName>
!# </surveyGeometryMethod>
```

Here, Geometry_Surveys_MARTIN_2010_ALFALFA_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:
17.4. Existing Method Types

subroutine Method_Initialize(surveyGeometryMethod, ,Geometry_Survey_Distance_Maximum_Get & &Geometry_Survey_Solid_Angle_Get ,Geometry_Survey_Volume_Maximum_Get, Geometry_Survey_Window_Functions_Get)
    implicit none
    type (varying_string ), intent(in ) :: surveyGeometryMethod
    procedure(double precision), pointer, intent(inout) :: Geometry_Survey_Distance_Maximum_Get, Geometry_Survey_Solid_Angle_Get & &Geometry_Survey_Volume_Maximum_Get
    procedure( ), pointer, intent(inout) :: Geometry_Survey_Window_Functions_Get
    if (surveyGeometryMethod == 'myMethod') then
        Geometry_Survey_Distance_Maximum_Get => My_Geometry_Survey_Distance_Maximum
        Geometry_Survey_Solid_Angle_Get => My_Geometry_Survey_Solid_Angle
        Geometry_Survey_Volume_Maximum_Get => My_Geometry_Survey_Volume_Maximum
        Geometry_Survey_Window_Functions_Get => My_Geometry_Survey_Window_Functions
    .
    .
    end if
    return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the surveyGeometryMethod input parameter. The procedure pointers must be set to point to functions which return properties of the survey as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The maximum distance function must have the form:

    double precision function Geometry_Survey_Distance_Maximum(mass)
    implicit none
    double precision, intent(in ) :: mass
    .
    .
    return
end function Geometry_Survey_Distance_Maximum

and must return the maximum distance (in Mpc) at which a galaxy of the given mass would be included in the survey.

The solid angle function must have the form:

    double precision function Geometry_Survey_Solid_Angle()
    implicit none
    .
    .
    return
end function Geometry_Survey_Solid_Angle

and should return the solid angle (in steradians) of the survey angular mask.

The maximum volume function must have the form:

    double precision function Geometry_Survey_Volume_Maximum(mass)
    implicit none
17. Adding New Methods

```fortran
double precision, intent(in) :: mass

return
end function Geometry_Survey_Volume_Maximum
```

and should return the maximum volume (in Mpc$^3$) within which a galaxy of the given mass could be found in the survey.

The window function function must have the form:

```fortran
double precision function Geometry_Survey_Window_Functions(mass1,mass2,boxLength,gridCount,windowFunction1,windowFunction2)
implicit none
double precision , intent(in) :: mass1,mass2
integer , intent(in) :: gridCount
double precision , intent(out) :: boxLength
complex(c_double_complex), intent(out), dimension(gridCount,gridCount,gridCount) :: windowFunction1,windowFunction2

return
end function Geometry_Survey_Window_Functions
```

and should compute the Fourier space window functions corresponding to the survey volume within which galaxies of masses mass1 and mass2 could be found. The window functions should be computed on a Cartesian grid, containing gridCount elements in each dimension. The chosen length of the box within which the survey geometry was embedded should be returned in the boxLength variable.

Currently defined type Ia supernovae methods are:

- Martin-2010-ALFALFA Computes the survey geometry of the survey of Martin et al. [2010].
- Li-White-2009-SDSS Computes the survey geometry of the survey of Li and White [2009].

**Tree Timing**

Additional methods for tree timing (i.e. the time taken to process a given merger tree) can be added using the `timePerTreeMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the file method is described by a directive:

```fortran
!# <timePerTreeMethod>
!# <unitName>Galacticus_Time_Per_Tree_File_Initialize</unitName>
!# </timePerTreeMethod>
```

Here, `Galacticus_Time_Per_Tree_File_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Method_Initialize(timePerTreeMethod,Galacticus_Time_Per_Tree_Get)
imPLICIT none
type(varying_string), intent(in) :: timePerTreeMethod
procedure(), pointer, intent(inout) :: Galacticus_Time_Per_Tree_Get

if (timePerTreeMethod == 'myMethod') then
```

846
17.4. Existing Method Types

Galacticus_Time_Per_Tree_Get => My_Time_Per_Tree_Get

end if
return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the timePerTreeMethod input parameter. The procedure pointer Galacticus_Time_Per_Tree_Get must be set to point to a function which returns an estimate of the time taken (in seconds) to process a merger tree. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The function must have the form:

    double precision function Time_Per_Tree(treeRootMass)
       implicit none
       double precision, intent(in) :: treeRootMass
    .
    .
    .
    return
    end function Time_Per_Tree

The function must return an estimate of the time taken (in seconds) to process a merger tree with the given treeRootMass.

Currently defined tree timing methods are:

file This method reads coefficients of a simple fitting formula for the processing time from a file, specified via the [timePerTreeFitFileName] parameter (see §18.2).

Transfer Function

Additional methods for transfer function can be added using the transferFunctionMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the file method is described by a directive:

!# <transferFunctionMethod>
!# <unitName>Transfer_Function_File_Initialize</unitName>
!# </transferFunctionMethod>

Here, Transfer_Function_File_Initialize is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

    subroutine Method_Initialize(transferFunctionMethod,Transfer_Function_Tabulate)
       implicit none
       type  (varying_string),  intent(in ) :: transferFunctionMethod
       procedure(      ), pointer, intent(inout) :: Transfer_Function_Tabulate
       procedure(      ), pointer, intent(inout) :: Transfer_Function_Half_Mode_Mass
    if (transferFunctionMethod == 'myMethod') then
       Transfer_Function_Tabulate => My_Do_Tabulate
       Transfer_Function_Half_Mode_Mass => My_Do_Half_Mode_Mass
    .
17. Adding New Methods

.

end if
return
end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the transferFunctionMethod input parameter. The procedure pointer transferFunction_Tabulate must be set to point to a subroutine which tabulates the transfer function as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The tabulation subroutine must have the form:

subroutine Transfer_Function_Tabulate(wavenumber,transferFunctionNumberPoints,transferFunctionWavenumber,transferFunctionT)
implicit none
double precision, intent(in) :: wavenumber
double precision, allocatable, dimension(:), intent(inout) :: transferFunctionLogWavenumber,transferFunctionLogT
integer, intent(out) :: transferFunctionNumberPoints
.
.
return
end subroutine Transfer_Function_Tabulate

The subroutine must tabulate the natural log of the transfer function in array transferFunctionLogT() as a function of the natural log of wavenumber transferFunctionLogWavenumber() (these arrays must be allocated to the correct size, and may be previously allocated, therefore requiring a deallocation). The number of tabulated points should be returned in transferFunctionNumberPoints. The subroutine should ensure that the currently requested wavenumber is within the range of the tabulated function (preferably with some buffer).

The half-mode mass function must have the form:

double precision function Transfer_Function_Half_Mode_Mass()
implicit none
.
.
return
end function Transfer_Function_Half_Mode_Mass

The function should return the mass, \( M_{hm} = \left(4\pi/3\right)(\Pi/k_{hm})^3 \), (in units of \( M_{\odot} \)) corresponding to the wavenumber, \( k_{hm} \), at which the transfer function is reduced by a factor 2 relative to the CDM case by small-scale dark matter particle physics.

Currently defined transfer function methods are:

null \( T(k) = 1 \).

file The transfer function is read from an XML file specified by input parameter transferFunctionFile.

CAMB The transfer function is generated by CAMB using the specified cosmological parameters. The transfer function is written out to a file in the data/ directory and will be re-read later if needed.

BBKS The transfer function is computed using the fitting formula of Bardeen et al. [1986].
Eisenstein-Hu1999 The transfer function is computed using the fitting formula of Eisenstein and Hu [1999]. The effective number of neutrino species and the summed mass (in electron volts) of all neutrino species are specified via the `effectiveNumberNeutrinos` and `summedNeutrinoMasses` parameters respectively.

The XML file format for transfer functions looks like:

```
<data>
  <column>k [Mpc^-1] - wavenumber</column>
  <column>T(k) - transfer function</column>
  <datum>1.111614e-05 0.218866E+08</datum>
  <datum>1.228521e-05 0.218866E+08</datum>
  <datum>1.357727e-05 0.218866E+08</datum>
  <datum>1.50052e-05 0.218866E+08</datum>
  <datum>1.658335e-05 0.218866E+08</datum>
  <datum>1.83274e-05 0.218865E+08</datum>
  ...
  ...
  ...
  <description>Cold dark matter power spectrum created by CAMB.</description>
  <fileFormat>1</fileFormat>
  <parameter>
    <name>Omega_b</name>
    <value>0.0450</value>
  </parameter>
  <parameter>
    <name>Omega_Matter</name>
    <value>0.250</value>
  </parameter>
  <parameter>
    <name>Omega_DE</name>
    <value>0.750</value>
  </parameter>
  <parameter>
    <name>H_0</name>
    <value>70.0</value>
  </parameter>
  <parameter>
    <name>T_CMB</name>
    <value>2.780</value>
  </parameter>
  <parameter>
    <name>Y_He</name>
    <value>0.24</value>
  </parameter>
  <extrapolation>
    <wavenumber>
      <limit>low</limit>
      <method>power law</method>
    </wavenumber>
  </extrapolation>
</data>
```
17. Adding New Methods

The `datum` elements give wavenumber (in Mpc$^{-1}$) and transfer function pairs. The `extrapolation` element defines how the tabulated function should be extrapolated to lower and higher wavenumbers. The two options for the `method` are “fixed”, in which case the transfer function is extrapolated assuming that it remains constant, and “power law” in which case the extrapolation is performed assuming a fixed power-law relation between transfer function and wavenumber. The `column`, `description` and `parameter` elements are optional, but are encouraged to make the file easier to understand. Finally, the `fileFormat` element should currently always contain the value 1—this may change in future if the format of this file is modified.

17.4.2. Events

Events are triggered during merger tree evolution. Examples are when a node needs to be promoted to its parent node, or when a minor node merges with its parent.

Node Merger Events

Additional methods for the node merging (i.e. when a non-primary progenitor merges with its parent) can be added using the `nodeMergersMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `singleLevelHierarchy` method is described by a directive:

```
!# <nodeMergersMethod>
!# <unitName>Events_Node_Merger.Initialize_SLH</unitName>
!# </nodeMergersMethod>
```

Here, `Satellite_Time_Until_Merging_Lacey_Cole_Initialize` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```fortran
subroutine Events_Node_Merger_Initialize(nodeMergersMethod,Events_Node_Merger_Do)
    implicit none
    type(varying_string), intent(in) :: nodeMergersMethod
    procedure(), pointer, intent(inout) :: Events_Node_Merger_Do

    if (nodeMergersMethod.eq.'myMethod') Events_Node_Merger_Do => My_Method_Procedure

    return
end subroutine Events_Node_Merger_Initialize
```

where `myMethod` is the name of this method as will be specified by the `nodeMergersMethod` input parameter. The procedure pointer `Events_Node_Merger_Do` must be set to point to a subroutine which handles the merging event as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The node merging subroutine must have the form:

```fortran
subroutine Events_Node_Merger_Do(thisNode)
    implicit none
```
17.4. Existing Method Types

The function must perform any processing required for the merger, and move `thisNode` to the linked list of satellite nodes in `thisNode%parentNode`.

Currently defined node merger event methods are:

**singleLevelHierarchy**  The node merger is handled by placing the merging node into the linked list of satellites of the parent node. Any satellites in the merging node are also promoted to be satellites in the new node, thereby maintaining just a single hierarchy level of substructure.

### Node Promotion Events

Additional methods for node promotion (i.e. when a primary progenitor reaches its parent halo) can be added using the `nodePromotionTask` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `basic` tree node method uses this directive as follows:

```plaintext
!# <nodePromotionTask>
!# <unitName>Tree_Node_Basic_Promote</unitName>
!# </nodePromotionTask>
```

Here, `Tree_Node_Basic_Promote` is the name of a subroutine which will be called to perform whatever tasks are required prior to the promotion. The subroutine must have the following form:

```plaintext
subroutine Node_Promotion_Task(thisNode)
  implicit none
  type(treeNode), pointer, intent(inout) :: thisNode
  .
  .
  return
end subroutine Node_Promotion_Task
```

where `thisNode` is the node about to be promoted.

### 17.4.3. Tasks

Tasks are any processing which must be performed on a node as a result of some specific event (such as a merger).

### Calculation Reset Tasks

Additional methods for calculation reset tasks (i.e. flagging that the properties of a node may have changed so that any calculations must be performed anew) can be added using the `calculationResetTask` directive. The directive should contain a single argument, giving the name of a subroutine to be called to perform the task. For example, the standard hot halo component adds a task as follows:
17. Adding New Methods

Here, **Tree_Node_Hot_Halo_Reset_Standard** is the name of a subroutine which will be called to perform whatever tasks are required. The subroutine must have the following form:

```fortran
subroutine Reset_Task(thisNode)
  implicit none
  type(treeNode), pointer, intent(inout) :: thisNode
  .
  .
  return
end subroutine Prederivative_Task
```

where `thisNode` is the node for which derivatives will be computed. Tasks typically involve precomputing quantities that will be used in finding the derivatives or resetting the state so that stored quantities will be recomputed as needed.

**Cooling Rate Modifiers**

Additional methods for modifying the cooling rate from the hot halo can be added using the `coolingRateModifierMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to modify the cooling rate. For example, the “cut-off” modifier is described by a directive:

```fortran
!# <coolingRateModifierMethod>
!# <unitName>Cooling_Rate_Modifier_Cut_Off</unitName>
!# </coolingRateModifierMethod>
```

Here, **Cooling_Rate_Modifier_Cut_Off** is the name of the subroutine which will be called modify the cooling rate. The modification subroutine must have the following form:

```fortran
subroutine Modify_Rate(thisNode,coolingRate)
  implicit none
  type(treeNode) , intent(inout), pointer :: thisNode
  double precision, intent(inout) :: coolingRate

  return
end subroutine Modify_Rate
```

The subroutine should modify the `coolingRate` as necessary. Currently defined cooling rate modifier tasks are:

- **“cut-off”**  The cooling rate is set to zero in halos with virial velocities below `[coolingCutOffVelocity]` at redshifts below/above `[coolingCutOffRedshift]` for `[coolingCutOffWhen]=after/before. In other halos the cooling rate is not modified.

**Decode Property Identifier Tasks**

Additional property identifier decoding tasks (i.e. determining the name of a property from a set of integer identifiers) can be added using the `decodePropertyIdentifiersTask` directive. The directive should contain a single argument, giving the name of a subroutine to be called to perform the task. For example, the Hernquist spheroid component adds a task as follows:
17.4. Existing Method Types

Here, Hernquist_Spheroid_Property_Identifiers_Decode is the name of a subroutine which will be called to perform the decoding task. The subroutine must have the following form:

```fortran
subroutine Property_Identifier_Decode_Task(propertyComponent,propertyObject,propertyIndex,matchedProperty,propertyName)
  implicit none
  integer, intent(in) :: propertyComponent,propertyObject,propertyIndex
  logical, intent(inout) :: matchedProperty
  type(varying_string), intent(inout) :: propertyName

return
end subroutine Property_Identifier_Decode_Task
```

The task should check whether propertyComponent matches its stored componentIndex value. If it does, it should set propertyName to a suitable name (e.g. hernquistSpheroid::stellarMass) and set matchedProperty = true. The value of propertyObject will be either objectTypeProperty indicating that the object in question is a standard property, or objectTypeHistory indicating that it is a history. The value of propertyIndex then gives the position of the object in question in the array of properties or histories.

**Evolution Timestep Tasks**

Merger tree nodes are evolved over some fixed timestep before evolution is stopped and other processing is allowed. The timestep is always sufficiently small such that the node does not evolve past the time of its parent node, nor does it evolve past the time of any of its satellite nodes. An arbitrary number of other criteria can be used to adjust the timestep. Such a criterion can be added using the timeStepsTask directive. For example, the simple timestep task adds itself using

```fortran
!# <timeStepsTask>
!# <unitName>Merger_Tree_Timestep_Simple</unitName>
!# </timeStepsTask>
```

Here, unitName gives the name of the subroutine to be called to (possibly) adjust the timestep. It should have the following form:

```fortran
subroutine My_Timestep(thisNode,timeStep,End_Of_Timestep_Task,report,lockNode,lockType)
  implicit none
  type (treeNode ), intent(inout), pointer :: thisNode
  procedure( ), intent(inout), pointer :: End_Of_Timestep_Task
  double precision , intent(inout) :: timeStep
  logical , intent(in ) :: report
  type (treeNode ), intent(inout), pointer, optional :: lockNode
  type (varying_string), intent(inout), optional :: lockType

return
end subroutine My_Timestep
```
17. Adding New Methods

This subroutine should compute a suitable timestep for thisNode and, if it is less than the currently defined value of timeStep should set timeStep to that value. Optionally, the procedure pointer End_Of_Timestep_Task can be set to point to a subroutine which will be called after the node is evolved to the end of the timestep. It is acceptable for this pointer to be null. Note that the End_Of_Timestep_Task will only be called for the task which provided the shortest timestep—other tasks can always request to be called again when the next timestep is determined. The subroutine to be called at the end of the timestep must have the form:

```
subroutine My_End_Of_Timestep_Task(thisTree,thisNode,deadlockStatus)
  implicit none
  type(mergerTree), intent(in) :: thisTree
  type(treeNode), intent(inout), pointer :: thisNode
  integer, intent(inout) :: deadlockStatus
  .
  return
end subroutine My_End_Of_Timestep_Task
```

The deadlockStatus argument should be set to isNotDeadlocked (provided by the Merger_Trees-Evolve_Deadlock_Status module) if, and only if, the end of timestep task makes some change to the state of the tree (e.g. merging a node), to indicate that the tree was not deadlocked in this pass (i.e. something actually changed in the tree).

If the report argument is true then the function should report the value of timestep prior to exiting. (This is used in reporting on timestepping criteri in deadlocked trees.) It is recommended that the report be made using the Evolve_To_Time_Report() function. Additionally, if the optional lockNode and lockType arguments are present then additional information can be supplied to aid in diagnosing deadlock conditions. If the current task is limiting the timestep then the lockNode pointer should be set to point to whichever node is causing the limit (which may be thisNode or some other node, e.g. a satellite of thisNode, etc.), and lockType should be set to a short description label identifying the type of limit.

Galactic Component Density

The function Galactic_Structure_Density() computes the density of material at a given position within a node. To have their density counted, each component must register a task using:

```
!# <densityTask>
!# <unitName>Density_Procedure</unitName>
!# </densityTask>
```

where Density_Procedure is the name of a function with the following template

```
double precision function Density_Procedure(thisNode,position,coordinateSystem,componentType,massType,weightBy,weightIndex,haloLoaded)
  type(treeNode), intent(inout), pointer :: thisNode
  integer, intent(in) :: massType,coordinateSystem,componentType,weightBy,weightIndex
  double precision, intent(in) :: radius
  logical , intent(in), optional :: haloLoaded
  .
  return
end function Density_Procedure
```
17.4. Existing Method Types

If \texttt{componentType} is a match to the component then the function should return the density of the component matching type \texttt{massType} at position for \texttt{thisNode}. \texttt{componentType} and \texttt{massType} can take one of the values described in §17.2. In the above “density” can actually refer to different quantities depending on the values of \texttt{weightBy} (and \texttt{weightIndex}): 

\texttt{weightByMass} The actual mass should be returned (the value of \texttt{weightIndex} is irrelevant); 

\texttt{weightByLuminosity} The \texttt{weightIndex}th luminosity should be returned.

If \texttt{haloLoaded=\text{true}} (which should be the default if this option is not present), then the effects of baryonic loading on the halo profile should be taken into account where necessary. Otherwise, the effects of baryonic loading should be ignored.

**Galactic Component Enclosed Mass**

The function \texttt{Galactic\_Structure\_Enclosed\_Mass()} computes the mass within a specified radius in a node. To have their mass counted, each component must register a task using:

\begin{verbatim}
!# <enclosedMassTask>
!#   <unitName>Enclosed_Mass_Procedure</unitName>
!# </enclosedMassTask>
\end{verbatim}

where \texttt{Enclosed\_Mass\_Procedure} is the name of a function with the following template

\begin{verbatim}
double precision function Enclosed\_Mass\_Procedure(thisNode,radius,componentType,massType,weightBy,weightIndex,haloLoaded)
    type(treeNode), intent(inout), pointer :: thisNode
    integer, intent(in) :: massType,componentType,weightBy,weightIndex
    double precision, intent(in) :: radius
    logical, intent(in), optional :: haloLoaded
    .
    .
    return
end function Enclosed\_Mass\_Procedure
\end{verbatim}

If \texttt{componentType} is a match to the component then the function should return the “mass” of the component matching type \texttt{massType} within \texttt{radius} for \texttt{thisNode}. \texttt{componentType} and \texttt{massType} can take one of the values described in §17.2. If \texttt{radius} is equal to or greater than \texttt{radiusLarge} the routine should return the total “mass” (i.e. “mass” within infinite radius). In the above “mass” can actually refer to different quantities depending on the values of \texttt{weightBy} (and \texttt{weightIndex}): 

\texttt{weightByMass} The actual mass should be returned (the value of \texttt{weightIndex} is irrelevant); 

\texttt{weightByLuminosity} The \texttt{weightIndex}th luminosity should be returned.

If \texttt{haloLoaded=\text{true}} (which should be the default if this option is not present), then the effects of baryonic loading on the halo profile should be taken into account where necessary. Otherwise, the effects of baryonic loading should be ignored.

**Galactic Component Rotation Curve**

The function \texttt{Galactic\_Structure\_Rotation\_Curve()} computes the rotation curve at a specified radius in a node. To have their contribution counted, each component must register a task using:
17. Adding New Methods

```fortran
!# <rotationCurveTask>
!# <unitName>Rotation_Curve_Procedure</unitName>
!# </rotationCurveTask>

where Rotation_Curve_Procedure is the name of a function with the following template

double precision function Rotation_Curve_Procedure(thisNode, radius, componentType, massType, haloLoaded)
  type(treeNode), intent(inout), pointer :: thisNode
  integer, intent(in) :: massType, componentType
  double precision, intent(in) :: radius
  logical , intent(in), optional :: haloLoaded
  .
  .
  .
  return
end function Rotation_Curve_Procedure

If componentType is a match to the component then the procedure should return the contribution to the rotation curve due to the component matching type massType within radius for thisNode.

Galactic Component Rotation Curve Gradient

The function Galactic_Structure_Rotation_Curve_Gradient() computes the gradient of the rotation curve at a specified radius in a node. To have their contribution counted, each component must register a task using:

```fortran
!# <rotationCurveGradientTask>
!# <unitName>Rotation_Curve_Gradient_Procedure</unitName>
!# </rotationCurveGradientTask>

where Rotation_Curve_Gradient_Procedure is the name of a function with the following template

double precision function Rotation_Curve_Gradient_Procedure(thisNode, radius, componentType, massType, haloLoaded)
  type(treeNode), intent(inout), pointer :: thisNode
  integer, intent(in) :: massType, componentType
  double precision, intent(in) :: radius
  logical , intent(in), optional :: haloLoaded
  .
  .
  .
  return
end function Rotation_Curve_Gradient_Procedure

If componentType is a match to the component then the function should return the contribution to the gradient of $V^2(r)$ due to the component matching type massType within radius for thisNode. Note that this is the gradient of the square of the rotation curve to permit gradients to be directly summed. componentType and massType can take one of the values described in §17.2. If haloLoaded=true (which should be the default if this option is not present), then the effects of baryonic loading on the halo profile should be taken into account where necessary. Otherwise, the effects of baryonic loading should be ignored.
Galactic Component Potential

The function `Galactic_Structure_Potential()` computes the potential at a specified radius in a node. To have their contribution counted, each component must register a task using:

```fortran
!# <potentialTask>
!# <unitName>Potential_Task</unitName>
!# </potentialTask>
```

where `Potential_Task` is the name of a function with the following template

```fortran
double precision function Potential_Procedure(thisNode, radius, componentType, massType, haloLoaded)
  type(treeNode), intent(inout), pointer :: thisNode
  integer, intent(in), optional :: componentType
  double precision, intent(in) :: radius
  logical , intent(in), optional :: haloLoaded
.
  return
end function Potential_Procedure
```

If `componentType` is a match to the component then the procedure should return the contribution to the rotation curve due to the component matching type `massType` within radius for `thisNode`. `componentType` and `massType` can take one of the values described in §17.2. If `haloLoaded`=true (which should be the default if this option is not present), then the effects of baryonic loading on the halo profile should be taken into account where necessary. Otherwise, the effects of baryonic loading should be ignored.

Galactic Component Surface Density

The function `Galactic_Structure_Surface_Density()` computes the surface density of material at a given position within a node. Note that while a 3-D position is specified the routine should return the surface density corresponding to integrating the component density through the minor axis (typically the z-axis). To have their surface density counted, each component must register a task using:

```fortran
!# <surfaceDensityTask>
!# <unitName>Surface_Density_Procedure</unitName>
!# </surfaceDensityTask>
```

where `Surface_Density_Procedure` is the name of a function with the following template

```fortran
double precision function Surface_Density_Procedure(thisNode, position, coordinateSystem, componentType, massType)
  type(treeNode), intent(inout), pointer :: thisNode
  integer, intent(in) :: massType, coordinateSystem, componentType
  double precision, dimension(3):: position
  logical , intent(in), optional :: haloLoaded
.
  return
end function Surface_Density_Procedure
```
17. Adding New Methods

If `componentType` is a match to the component then the function should return the surface density of the component matching type `massType` at `position` for `thisNode`. `componentType` and `massType` can take one of the values described in §17.2. The coordinate system in which `position` is specified is given by `coordinateSystem` which can take on the following values:

- `coordinateSystemCartesian` Cartesian \((x, y, z)\);
- `coordinateSystemSpherical` Spherical \((r, \theta, \phi)\);
- `coordinateSystemCylindrical` Cylindrical \((R, \phi, z)\).

If `haloLoaded` = `true` (which should be the default if this option is not present), then the effects of baryonic loading on the halo profile should be taken into account where necessary. Otherwise, the effects of baryonic loading should be ignored.

### Halo Formation Events

Tasks to be performed when a halo is deemed to have “formed” (or reformed) can be registered using the `haloFormationTask` directive. For example, the `Tree_Node_Methods_Hot_Halo` module registers a task using

```
!# <haloFormationTask>
!# <unitName>Hot_Halo_Formation_Task</unitName>
!# </haloFormationTask>
```

The contents of `<unitName>` should give the name of the subroutine to be called on halo formation. The subroutine should have a single argument, `thisNode`, which is the node that has (re)formed.

### HDF5 File Close

Tasks to be performed just prior to closing the GALACTICUS output HDF5 file (typically involving writing accumulated data to that file) can be registered using the `hdfPreCloseTask` directive. For example, the `Merger_Tree_Timesteps_History` module registers a task using

```
!# <hdfPreCloseTask>
!# <unitName>Merger_Tree_History_Write</unitName>
!# </hdfPreCloseTask>
```

The contents of `<unitName>` should give the name of the subroutine to be called prior to HDF5 file closure. The subroutine should have no arguments.

### Initial Mass Functions

New IMF s can be added using the `imfRegister` and `imfRegisterName` task directives. For example, the Salpeter IMF is registered using the directives:

```
!# <imfRegister>
!# <unitName>Star_Formation_IMF_Register_Salpeter</unitName>
!# </imfRegister>
```

and

```
!# <imfRegisterName>
!# <unitName>Star_Formation_IMF_Register_Name_Salpeter</unitName>
!# </imfRegisterName>
```
17.4. Existing Method Types

The `unitName` tags specify subroutines that are called to register the IMF. These subroutines should have the following forms:

```fortran
subroutine Star_Formation_IMF_Register_My_IMF(imfAvailableCount)
    integer, intent(inout) :: imfAvailableCount

    imfAvailableCount=imfAvailableCount+1
    myImfIndex       =imfAvailableCount

    return
end subroutine Star_Formation_IMF_Register_My_IMF

subroutine Star_Formation_IMF_Register_Name_My_IMF(imfNames,imfDescriptors)
    type(varying_string), intent(inout), dimension(:) :: imfNames,imfDescriptors

    imfNames (myImfIndex)="Salpeter"
    imfDescriptors(myImfIndex)="Salpeter"

    return
end subroutine Star_Formation_IMF_Register_Name_My_IMF
```

The first routine should increment the `imfAvailableCount` counter by 1 and keep a record of the resulting index—this will be the index by which the IMF is referred to. The second routine should store the name and descriptor of the IMF in the appropriate position in the supplied `imfNames()` and `imfDescriptors()` arrays. The “name” is the label used to identify the IMF in input parameters for example. The “descriptor” should be a label sufficient to uniquely identify the IMF, and is used, for example, in constructing file names when storing IMF related data. Often, the name and descriptor are identical. However, if the IMF has user-definable parameters then those parameters should be encoded into the descriptor.

Each registered IMF should supply a set of functions as described in §17.4.1.

### Merger Tree Extra Output Tasks

Extra outputs for merger trees (i.e. those which do not involve output of a fixed number of properties for every node—examples might be star formation histories for a subset of galaxies) can be added using the directive: `mergerTreeExtraOutputTask`. The directive should give the name of the subroutine to be called to perform the task. A template for this task is:

```fortran
!# <mergerTreeExtraOutputTask>
!# <unitName>Galacticus_Extra_Output_Example</unitName>
!# </mergerTreeExtraOutputTask>
subroutine Galacticus_Extra_Output_Example(thisNode,iOutput,treeIndex,nodePassesFilter)
    implicit none
    type(treeNode), intent(inout), pointer :: thisNode
    integer, intent(in) :: iOutput
    integer(kind=kind_int8), intent(in) :: treeIndex
    logical, intent(in) :: nodePassesFilter

    .
    .
    return
end subroutine Galacticus_Extra_Output_Example
```

The subroutine will be called for each node in each merger tree at each output, and should perform whatever extra output related to `thisNode`. The index of the output and tree are provided as `iOutput`
and `treeIndex` for reference, and may be used in organizing output. The `nodePassesFilter` flag will be set to `true` if `thisNode` passed all active output filters (see §17.4.3). If it is `false` then typically no output should occur (although other tasks may still be undertaken).

**Merger Tree Output Tasks**

Additional outputs for merger trees can be added using three directives: `mergerTreeOutputPropertyCount`, `mergerTreeOutputNames` and `mergerTreeOutputTask`. Each directive should give the name of the subroutine to be called to perform the task and, additionally, a name for sorting (this should be the same for all three directives and ensures that output tasks are always called in the correct order). Templates for these tasks are:

```
!# <mergerTreeOutputNames>
!# <unitName>Galacticus_Output_Tree_Example_Names</unitName>
!# <sortName>Galacticus_Output_Tree_Example</sortName>
!# </mergerTreeOutputNames>
subroutine Galacticus_Output_Tree_Example_Names(integerProperty,integerPropertyNames,integerPropertyComments,integerPropertyUnitsSI &
   &,doubleProperty,doublePropertyNames,doublePropertyComments,doublePropertyUnitsSI,time)
   implicit none
   double precision, intent(in) :: time
   integer, intent(inout) :: integerProperty,doubleProperty
   character(len=*) , intent(inout), dimension(:) :: integerPropertyNames,integerPropertyComments,doublePropertyNames &
   & ,doublePropertyComments
   double precision, intent(inout), dimension(:) :: integerPropertyUnitsSI,doublePropertyUnitsSI
   .
   .
   return
end subroutine Galacticus_Output_Tree_Example_Names

!# <mergerTreeOutputPropertyCount>
!# <unitName>Galacticus_Output_Tree_Example_Property_Count</unitName>
!# <sortName>Galacticus_Output_Tree_Example</sortName>
!# </mergerTreeOutputPropertyCount>
subroutine Galacticus_Output_Tree_Example_Property_Count(integerPropertyCount,doublePropertyCount)
   implicit none
   integer, intent(inout) :: integerPropertyCount,doublePropertyCount
   .
   .
   return
end subroutine Galacticus_Output_Tree_Example_Property_Count

!# <mergerTreeOutputTask>
!# <unitName>Galacticus_Output_Tree_Example</unitName>
!# <sortName>Galacticus_Output_Tree_Example</sortName>
!# </mergerTreeOutputTask>
subroutine Galacticus_Output_Tree_Example(thisNode,integerProperty,integerBufferCount,integerBuffer,doubleProperty &
   &,doubleBufferCount,doubleBuffer)
   implicit none
17.4. Existing Method Types

```fortran
  type(treeNode), intent(inout), pointer :: thisNode
  integer, intent(inout) :: integerProperty, integerBufferCount, doubleProperty, doubleBufferCount
  integer(kind=kind_int8), intent(inout) :: integerBuffer(:,:)
  double precision, intent(inout) :: doubleBuffer(:,:)

  !
  !
  !
  return
end subroutine Galacticus_Output_Tree_Example
```

The `mergerTreeOutputPropertyCount` subroutine must simply increment `integerPropertyCount` and `doublePropertyCount` by the number of integer and double precision properties that will be output respectively. The `mergerTreeOutputNames` subroutine must store the dataset names, comments and units in the SI system\(^9\) for each integer and double precision property in the supplied arrays. The value of `integerProperty` and `doubleProperty` should be incremented by 1 before each property name/comment is set—these then supply the position within the input arrays in which to store the name. The `mergerTreeOutputTask` subroutine must similarly place the desired property values for `thisNode` into the supplied arrays. The value of `integerProperty` and `doubleProperty` should be incremented by 1 before each property value is set. The value can then be stored in, for example, `integerBuffer(integerBufferCount,integerProperty)`.

### Merger Tree Pre-Construction Tasks

Additional tasks to be performed prior to the construction of each merger tree can be added using the `mergerTreePreTreeConstructionTask` directive. For example, the tree timing task uses this directive as follows:

```fortran
!# <mergerTreePreTreeConstructionTask>
!# <unitName>Meta_Tree_Timing_Pre_Construction</unitName>
!# </mergerTreePreTreeConstructionTask>
```

Here, `Meta_Tree_Timing_Pre_Construction` is the name of a subroutine which will be called to perform whatever tasks are required. The subroutine must have the following form:

```fortran
subroutine Merger_Tree_PreConstruction_Task()
  implicit none
  .
  .
  return
end subroutine Merger_Tree_PreConstruction_Task
```

The subroutine will be called once for each tree, before the tree has been constructed.

### Merger Tree Post-Evolution Tasks

Additional tasks to be performed after the evolution (and subsequent destruction) of each merger tree can be added using the `mergerTreePostEvolveTasker` directive. For example, the tree timing task uses this directive as follows:

```fortran
!# <mergerTreePostEvolveTasker>
!# <unitName>Meta_Tree_Timing_Post_Evolve</unitName>
!# </mergerTreePostEvolveTasker>
```

For dimensionless quantities, the units may be set to zero. In such cases, the `unitsInSI` attribute for the dataset will not be written to the `Galacticus` output file.

\(^9\)For dimensionless quantities, the units may be set to zero. In such cases, the `unitsInSI` attribute for the dataset will not be written to the `Galacticus` output file.
17. Adding New Methods

Here, Meta_Tree_Timing_Post_Evolve is the name of a subroutine which will be called to perform whatever tasks are required. The subroutine must have the following form:

```
subroutine Merger_Tree_PostEvolution_Task()
    implicit none
    .
    .
    return
end subroutine Merger_Tree_PostEvolution_Task
```

The subroutine will be called once for each tree, after the tree has been evolved and destroyed.

**Merger Tree Pre-Evolution Tasks**

Additional tasks to be performed on merger trees prior to their evolution can be added using the `mergerTreePreEvolveTask` directive. For example, the mass accretion history task uses this directive as follows:

```
!# <mergerTreePreEvolveTask>
!# <unitName>Merger_Tree_Mass_Accretion_History_Output</unitName>
!# </mergerTreePreEvolveTask>
```

Here, Merger_Tree_Mass_Accretion_History_Output is the name of a subroutine which will be called to perform whatever tasks are required. The subroutine must have the following form:

```
subroutine Merger_Tree_PreEvolution_Task(thisTree)
    implicit none
    type(mergerTree), intent(in) :: thisTree
    .
    .
    return
end subroutine Merger_Tree_PreEvolution_Task
```

where thisTree is the tree to be processed. The function will be called once for each tree, prior to the tree being evolved. Note that thisTree may link to other trees via its nextTree pointer. The function may want to process each tree in this linked list.

**Merger Tree Initialization Tasks**

Additional tasks to be performed during merger tree initialization can be added using the `mergerTreeInitializeTask` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the standard basic component method uses this directive as follows:

```
!# <satelliteMergerTask>
!# <unitName>Halo_Mass_Accretion_Rate</unitName>
!# </satelliteMergerTask>
```
Here, \texttt{Halo\_Mass\_Accretion\_Rate} is the name of a subroutine which will be called to perform whatever tasks are required as a result of the merger. The subroutine must have the following form:

```fortran
subroutine Merger\_Tree\_Initialize\_Task(thisNode)
    implicit none
    type(tree\_Node), pointer, intent(inout) :: thisNode
    .
    .
    return
end subroutine Merger\_Tree\_Initialize\_Task
```

where \texttt{thisNode} is the node to be initialized. The subroutine will be called once for each node in the tree.

**Merger Tree Structure Output Tasks**

Additional outputs for merger tree structure output can be added using the \texttt{mergerTreeStructureOutputTask}. The directive should give the name of the subroutine to be called to perform the task. The templates for this tasks is:

```fortran
!# <mergerTreeStructureOutputTask>
!# <unitName>Structure\_Output\_Task</unitName>
!# </mergerTreeStructureOutputTask>
subroutine Structure\_Output\_Task(baseNode,nodeProperty,treeGroup)
    implicit none
    type(tree\_Node), intent(in), pointer :: baseNode
double precision, intent(inout), dimension(:) :: nodeProperty
type(hdf5\_Object), intent(inout) :: treeGroup
    .
    .
    return
end subroutine Structure\_Output\_Task
```

The subroutine must walk the merger tree beginning from the given \texttt{baseNode} and store each property to output in the given \texttt{nodeProperty} array. Once populated, this array can be written to the appropriate HDF5 group, given by \texttt{treeGroup}, in the \texttt{GALACTICUS} output file.

**Node Dump**

The function \texttt{Node\_Dump(thisNode)} writes out all properties of a node to the display. To have their properties listed, each component must register a task using:

```fortran
!# <nodeDumpTask>
!# <unitName>Node\_Dump\_Procedure</unitName>
!# </nodeDumpTask>
```

where \texttt{Node\_Dump\_Procedure} is the name of a subroutine with the following template

```fortran
subroutine Node\_Dump\_Procedure(thisNode)
    type(tree\_Node), intent(inout), pointer :: thisNode
    .
```

863
17. Adding New Methods

return
end subroutine Node_Dump_Procedure

If the node contains an active component, this subroutine should display all relevant properties of the
component. If not, it can display a short message indicating that fact.

Output Filter Tasks

Extra filters for controlling which galaxies are output can be added using the directives \texttt{mergerTreeOutputFilterInitialize}
and \texttt{mergerTreeOutputFilter}. Each directive should give the name of the function to be called to ini-
tialize or apply the filter respectively. A template for these tasks is:

\begin{verbatim}
!# <mergerTreeOutputFilterInitialize>
!# <unitName>Galacticus_Merger_Tree_Output_Filter_Initialize_Example</unitName>
!# </mergerTreeOutputFilterInitialize>
subroutine Galacticus_Merger_Tree_Output_Filter_Initialize_Example(filterNames)
  implicit none
  type(varying_string), intent(in), dimension(:) :: filterNames
  .
  .
  return
end subroutine Galacticus_Merger_Tree_Output_Filter_Initialize_Example

!# <mergerTreeOutputFilter>
!# <unitName>Galacticus_Merger_Tree_Output_Filter_Example</unitName>
!# </mergerTreeOutputFilter>
logical function Galacticus_Merger_Tree_Output.Filter_Example(thisNode,doOutput)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  logical, intent(inout) :: doOutput
  .
  .
  return
end function Galacticus_Merger_Tree_Output_Filter_Example
\end{verbatim}

The initialization subroutine will be called prior to any use of the filter function. The \texttt{filterNames}
arrays contains a list of all filters which were requested to be applied. The function should check if its
filter is listed in this array and activate itself if necessary. The filter function will be called for each node,
\texttt{thisNode}, which is being considered for output. If the filter is activate, it should determine whether
\texttt{thisNode} passes its criteria for output. If it does not, \texttt{doOutput} should be set to false. If the output
criteria are met, then \texttt{doOutput should not be changed} (as it may already have been set false by some
other filter).

Currently available filters, selected using the [mergerTreeOutputFilters] input parameter, are:

\begin{itemize}
\item \texttt{lightcone} Restricts output to those galaxies which fall within a specified lightcone geometry. See §14.8 for
  further details;
\item \texttt{stellarMass} Restricts output to those galaxies with a total stellar mass equal to or greater than [stellarMassFilterThreshold];
\end{itemize}
17.4. Existing Method Types

luminosity  Restricts output to those galaxies with a total absolute AB magnitude less than or equal to \[^{10}\] \([\text{luminosityFilterAbsoluteMagnitudeThresholds}]\). This input parameter should be an array, with one entry for each luminosity being computed. The filter will be applied only for those luminosities that are being output at the current time.

### Output Group Output Tasks

Extra outputs for output groups (i.e. the groups which hold all merger tree data for a given output time) can be added using the directive: `outputGroupOutputTask`. The directive should give the name of the subroutine to be called to perform the task. A template for this task is:

```markdown
!# <outputGroupOutputTask>
!# <unitName>Galacticus_Output_Group_Output_Example</unitName>
!# </outputGroupOutputTask>

subroutine Galacticus_Output_Group_Output_Example(outputGroup,time)
  implicit none
  type(hdf5Object), intent(inout) :: outputGroup
  double precision, intent(in) :: time
  .
  .
  return
end subroutine Galacticus_Output_Group_Output_Example
```

The subroutine will be called for each output group created, and should perform whatever extra output it requires. The `outputGroup` object and the corresponding output `time` are provided as input parameters.

### Pre-derivative Tasks

Additional methods for pre-derivative tasks (i.e. things that should be done just prior to the computation of derivatives or properties for a node) can be added using the `preDerivativeTask` directive. The directive should contain a single argument, giving the name of a subroutine to be called to perform the task. For example, the standard hot halo component adds a task as follows:

```markdown
!# <preDerivativeTask>
!# <unitName>Tree_Node_Hot_Halo_Prederivative_Standard</unitName>
!# </preDerivativeTask>
```

Here, `Tree_Node_Hot_Halo_Prederivative_Standard` is the name of a subroutine which will be called to perform whatever tasks are required. The subroutine must have the following form:

```markdown
subroutine Prederivative_Task(thisNode)
  implicit none
  type(treeNode), pointer, intent(inout) :: thisNode
  .
  .
  return
end subroutine Prederivative_Task
```

where `thisNode` is the node for which derivatives will be computed. Tasks typically involve precomputing quantities that will be used in finding the derivatives.

[^10]: That is, galaxies which are at least as luminous as the specified threshold.
17. Adding New Methods

**Radius Solver Tasks**

Galactic radii solver functions (see §17.4.1) need to be able to interact with the components of a tree node to

1. Determine which components want a radius to be solved for;
2. Get and set the properties of those components.

The `radiusSolverPlausibility` and `radiusSolverTask` directives facilitate this. A component which has a radius to be solved for should include directives of the form:

```
!# <radiusSolverTask>
!# <unitName>Component_Radius_Solver_Plausibility</unitName>
!# </radiusSolverTask>
```

and

```
!# <radiusSolverTask>
!# <unitName>Component_Radius_Solver</unitName>
!# </radiusSolverTask>
```

where `Component_Radius_Solver_Plausibility` is the name of a subroutine which will specify whether or not the component is physically plausible for radius solving (e.g. has non-negative mass) and should have the following form:

```plaintext
subroutine Component_Radius_Solver_Plausibility(thisNode,galaxyIsPhysicallyPlausible)
  implicit none
  type(treeNode), pointer, intent(inout) :: thisNode
  logical,     intent(inout) :: galaxyIsPhysicallyPlausible
  .
  .
  .
  return
end subroutine Component_Radius_Solver_Plausibility
```

which should set `galaxyIsPhysicallyPlausible` to false if the component is not physically plausible, but should otherwise leave `galaxyIsPhysicallyPlausible` unchanged. Additionally, `Component_Radius_Solver` is the name of a subroutine which will supply the necessary information about the node, and which should have the following form:

```plaintext
subroutine Component_Radius_Solver(thisNode,componentActive,specificAngularMomentum,Radius_Get,Radius_Set,Velocity_Get,Velocity_Set)
  implicit none
  type(treeNode), pointer, intent(inout) :: thisNode
  logical,     intent(out) :: componentActive
  double precision,     intent(out) :: specificAngularMomentum
  procedure(),       pointer, intent(out) :: Radius_Get,Velocity_Get
  procedure(),       pointer, intent(out) :: Radius_Set,Velocity_Set
  .
  .
  return
end subroutine Component_Radius_Solver
```
When called, the subroutine should set `componentActive` to indicate whether or not this node contains an active component of the type. If it does, it should also set `specificAngularMomentum` to reflect the specific angular momentum (in km s\(^{-1}\) Mpc) of the component (at whatever point in its profile the radius is required) and should point the four procedure pointers to routines which get and set the radius and circular velocity properties of the component (which should have the standard form for component get and set methods). It is acceptable for the set procedures to point to dummy routines.

The galactic structure radii solver routines will use this information to determine (and set) the radius and circular velocity of the component. An advantage of this approach is that different radii solver methods can all use this same system, ensuring that just a single interface is needed in each component.

**Satellite Host Change Tasks**

Additional methods for satellite host change events (i.e. when a satellite node moves to a new host) can be added using the `satelliteHostChangeTask` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `simple` satellite orbits components uses this directive as follows:

```plaintext
! # <satelliteHostChangeTask>
! # <unitName>Satellite_Orbit_New_Host</unitName>
! # </satelliteHostChangeTask>
```

Here, `Satellite_Orbit_New_Host` is the name of a subroutine which will be called to perform whatever tasks are required as a result of the host change. The subroutine must have the following form:

```plaintext
subroutine New_Host_Task(thisNode)
  implicit none
  type(treeNode), pointer, intent(inout) :: thisNode
  .
  .
  return
end subroutine New_Host_Task
```

where `thisNode` is the node which has changed host (the new host halo is `thisNode%parentNode`).

**Satellite Merger Tasks**

Additional methods for satellite merger tasks can be added using the `satelliteMergerTask` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `simple` satellite orbits components uses this directive as follows:

```plaintext
! # <satelliteMergerTask>
! # <unitName>Satellite_Merger_Task</unitName>
! # </satelliteMergerTask>
```

Here, `Satellite_Merger_Task` is the name of a subroutine which will be called to perform whatever tasks are required as a result of the merger. The subroutine must have the following form:

```plaintext
subroutine Satellite_Merger_Task(thisNode)
  implicit none
  type(treeNode), pointer, intent(inout) :: thisNode
  .
  .
```
17. Adding New Methods

. return  
end subroutine Satellite_Merger_Task  

where thisNode is the node about to merge with thisNode%parentNode.

Star Formation History Tasks

Additional methods for star formation history tracking can be added using the starFormationHistoriesMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the metallicitySplit method uses this directive as follows:

!# <starFormationHistoriesMethod>  
!# <unitName>Star_Formation_Histories_Metallicity_Split_Initialize</unitName>  
!# </starFormationHistoriesMethod>

Here, Star_Formation_Histories_Metallicity_Split_Initialize is the name of a subroutine which will be called to initialize the method. The subroutine must have the following form:

```fortran
subroutine Method_Initialize( starFormationHistoriesMethod &,  
                           Star_Formation_History_Create_Do &,  
                           Star_Formation_History_Scales_Do &,  
                           Star_Formation_History_Record_Do &,  
                           Star_Formation_History_Output_Do & )  
implicit none  
type(varying_string), intent(in) :: starFormationHistoriesMethod  
procedure(), pointer, intent(inout) :: Star_Formation_History_Create_Do &,  
                                      Star_Formation_History_Scales_Do &,  
                                      Star_Formation_History_Record_Do &,  
                                      Star_Formation_History_Output_Do &  

if (starFormationHistoriesMethod == 'myMethod') then  
   Star_Formation_History_Create_Do => My_Create  
   Star_Formation_History_Scales_Do => My_Scales  
   Star_Formation_History_Record_Do => My_Record  
   Star_Formation_History_Output_Do => My_Output  
end if  
return  
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the starFormationHistoriesMethod input parameter. The procedure pointers must be set to point to subroutines which perform the functions described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The Star_Formation_History_Create_Do subroutine must have the form:

```fortran
subroutine My_Create(thisNode,thisHistory)  
implicit none  
type(treeNode), intent(inout), pointer :: thisNode  
type(history), intent(inout) :: thisHistory  
return  
end subroutine My_Create
```
17.4. Existing Method Types

and should return a history object in \texttt{thisHistory} suitable for holding a star formation history for \texttt{thisNode}.

The \texttt{Star\_Formation\_History\_Scales\_Do} subroutine must have the form:

```fortran
subroutine My_Scales(thisHistory, stellarMass, stellarAbundances)
  implicit none
  double precision, intent(in) :: stellarMass
  type(abundancesStructure), intent(in) :: stellarAbundances
  type(history), intent(inout) :: thisHistory
  return
end subroutine My_Scales
```

and should set the ODE solver error tolerance scales in \texttt{thisHistory}, using the provided information on \texttt{stellarMass} and \texttt{stellarAbundances} if required.

The \texttt{Star\_Formation\_History\_Record\_Do} subroutine must have the form:

```fortran
subroutine My_Record(thisNode, thisHistory, fuelAbundances, starFormationRate)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  type(history), intent(inout) :: thisHistory
  type(abundancesStructure), intent(in) :: fuelAbundances
  double precision, intent(in) :: starFormationRate
  return
end subroutine My_Record
```

and should record the contribution to the star formation history in \texttt{thisHistory} for \texttt{thisNode} given the current \texttt{starFormationRate} and star formation \texttt{fuelAbundances}. That is, the subroutine should adjust the rates in \texttt{thisHistory} appropriately.

The \texttt{Star\_Formation\_History\_Output\_Do} subroutine must have the form:

```fortran
subroutine My_Output(thisNode, thisHistory, iOutput, treeIndex, componentLabel)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  type(history), intent(inout) :: thisHistory
  integer, intent(in) :: iOutput
  integer(kind=kind_int8), intent(in) :: treeIndex
  character(len=*) :: componentLabel
  return
end subroutine My_Output
```

and should write the star formation history, \texttt{thisHistory}, for \texttt{thisNode} to the output file. The output number and tree index are provided as \texttt{iOutput} and \texttt{treeIndex} for reference, and \texttt{componentLabel} provides a suitable label for the component to which the history belongs (and so should be used in the name of the datasets to which the history is written for example).

Conventionally, star formation histories are output as follows:

```hdf5
"galacticus.hdf5" {
  GROUP "starFormationHistories" {
    COMMENT "Star formation history data."
    GROUP "Output1" {
      COMMENT "Star formation histories for all trees at each out"
      GROUP "mergerTree1" {
```

869
17. Adding New Methods

where \( \text{nodeID} \) is the index of the relevant node. The specifics of each dataset will depend on the selected star formation history method.

Currently defined star formation history methods are:

**metallicitySplit** The star formation history is tabulated on a grid of time and metallicity. The binning in time is chosen such that bins are at most of size \([\text{starFormationHistoryTimeStep}]\) between the time at which each galaxy formed and the final output time, and at most of size \([\text{starFormationHistoryFineTimeStep}]\) in the period \([\text{starFormationHistoryFineTime}]\) prior to each output time (all times specified in Gyr). The allows fine binning of recent star formation just prior to each output. The metallicity binning is arranged logarithmically in metallicity with \([\text{starFormationHistoryMetallicityCount}]\) bins between \([\text{starFormationHistoryMetallicityMinimum}]\) and \([\text{starFormationHistoryMetallicityMaximum}]\) (specified in Solar units). Note that the metallicity associated with each bin is the minimum metallicity for that bin (the maximum being the metallicity value associated with the next bin, except for the final bin which extends to infinite metallicity). If \([\text{starFormationHistoryMetallicityCount}] = 0\) is set, then the star formation
history is not split by metallicity (i.e. a single metallicity bin encompassing all metallicities from zero to infinity is used). Output follows the conventional format, with 2D star formation history datasets to represent the history as a function of time and metallicity. An additional `metallicities` dataset is added to the `starFormationHistories` output group to record the metallicity binning as follows:

```
DATASET "metallicities" {
    COMMENT "Metallicities at which star formation histories are tabulated"
    DATATYPE H5T_IEEE_F64LE
    DATASPACE SIMPLE { ( [starFormationHistoryMetallicityCount] ) / ( [starFormationHistoryMetallicityCount] ) }
}
```

### 17.5. Subsystems

This section describes some of the subsystems within GALACTICUS that support various physical entities or processes.

#### 17.5.1. Kepler Orbits

The `keplerOrbit` object (provided by the `Kepler_Orbits_Structure` module) stores the parameters of a single Keplerian orbit. It internally handles computation of additional/alternate orbital parameters once an orbit has been fully defined. Currently, the orientation of orbits (i.e. the unit vector normal to the orbital plane and the argument of periapsis) is not tracked. As such, orbits are fully defined by three parameters (in addition to the masses of the orbitting bodies). The following limitations presently apply to the `keplerOrbit` object:

- If an orbit is overdefined (i.e. if more than three parameters are set manually) no checking is performed to ensure that the parameters are consistent with a Keplerian orbit;
- Not all interconversions between parameters are supported\(^\text{11}\). If a conversion cannot be performed, an error message will be given.

A `keplerOrbit` object can be reset by calling the `reset()` method, and its defined/undefined status can be tested with the `isDefined()` method or asserted with the `assertIsDefined()` method. The following orbital parameters are supported, each method returning the value of the parameter and a corresponding method suffixed with `Set` can be used to set the parameter: `radius`, `velocityRadial`, `velocityTengential`, `energy`, `angularMomentum`, `eccentricity`, `semiMajorAxis`, `radiusPericenter`, `radiusApocenter`. Additionally, the masses of the orbitting bodies are provided by the `hostMass()` and `reducedMassSpecific()` = \( M_{\text{host}} / (M_{\text{host}} + M_{\text{satellite}}) \) methods. Finally, the `velocityScale()` method returns \( G M_{\text{host}} / r \) where \( r \) is the radius of the orbit.

#### 17.5.2. Chemicals

The chemicals subsystem provides both a interface to a database of known chemicals (allowing their physical properties to be queried) and a structure to store abundances/masses/etc. of the set of chemicals being tracked in GALACTICUS. The name “chemicals” is used to denote any chemical species that might be involved in reactions, including molecules, atoms, atomic and molecular ions and electrons.

\(^{11}\) The `keplerOrbit` object works by trying to convert to the combination radius, radial and tangential velocities. Once these are defined, all other parameters can be computed. However, for orbits defined in terms of other parameters, the `keplerOrbit` object does not know how to convert from every such combination of parameters.
Chemical Database

The file data/Chemical_Database.cml contains a database of chemicals that can currently be used by GALACTICUS. It uses a simplified version of the Chemical Markup Language to describe chemicals. An excerpt from the database is shown below:

```xml
<list>
  <chemical>
    <id>MolecularHydrogenAnion</id>
    <formalCharge>-1</formalCharge>
    <atomArray>
      <atom>
        <id>1</id>
        <elementType>H</elementType>
      </atom>
      <atom>
        <id>2</id>
        <elementType>H</elementType>
      </atom>
    </atomArray>
    <bondArray>
      <bond>
        <atomRefs2>1 2</atomRefs2>
        <order>1</order>
      </bond>
    </bondArray>
  </chemical>
</list>
```

The database contains a list of chemicals, each contained within a chemical element. The id element provides a label for the chemical (usually a descriptive name with no white space). The formalCharge element gives the charge of the chemical in units of the elementary charge. The chemical is then describe by a list of atoms and bonds inside atomArray and bondArray elements respectively. The atomArray can contain any number of atom elements, which should describe each atom in the chemical giving it a unique id number and an elementType, which is the short one or two letter label for the element (e.g. H, Ni, etc.). The bondArray should contain a bond entry for each atomic bond, which itself contains a atomRefs2 element giving the IDs of the two atoms participating in the bond and an order element which gives the order of the bond (e.g. “1” for a single bond).

Chemical Structure

Within GALACTICUS a chemical is represented using the chemicalStructure type which is provided by the Chemical_Structures module. A chemicalStructure object can be assigned a particular chemical by retrieving that chemical from the database using:

```python
call myChemical%retrieve("chemicalID")
```

where chemicalID is the ID of the chemical in the database. Any chemical can be exported to a CML file using
call myChemical%export(fileName)

where fileName gives the name of the file to which to export.

Once assigned a chemical, basic properties such as mass and charge (in atomic units) can be accessed using myChemical%mass and myChemical%charge respectively. The mass is computed from the known atomic masses of the constituent atoms of the chemical.

**Chemical Abundances**

Within GALACTICUS a set of abundances (or masses, or densities...) for all chemicals being tracked, as specified by the [chemicalsToTrack] input parameter, is stored within a chemicalAbundancesStructure type, as provided by the Chemical_Abundances_Structure module. The structure provides interfaces for setting and retrieving the abundance of a given chemical species, to pack/unpack all chemicals to/from an array, to convert from mass-weighted to number-weighted quantities and to multiply and divide the chemicals abundances by a given amount. Additionally, the Chemical_Abundances_Structure module provides functions which provide a count of the number of chemicals tracked, to look up the index of a chemical array representation from its name, and to retrieve the name of a given chemical.

**17.5.3. Radiation**

This subsystem handles radiation fields, providing convenient means to communicate radiation fields from one part of the GALACTICUS code to another. A radiation object can hold multiple different types of radiation field (e.g. it could contain both the cosmic microwave background and an interstellar radiation field localized to a specific galaxy).

**Radiation Structure**

Within GALACTICUS radiation fields are represented by the radiationStructure type which is provided by the Radiation_Structures module. A radiationStructure object must first be defined using:

    call myRadiation%define([radiationType1,radiationType2])

where the list of radiationTypes specifies what radiation components will be present in this radiation object. Currently defined radiation types are:

- **CMB** The cosmic microwave background;
- **Null** A null (zero radiation) component.

For example,

    call myRadiation%define([radiationTypeCMB])

will define the myRadiation object to contain just the cosmic microwave background.

Once defined, the specific radiation field can be set using:

    call myRadiation%set(thisNode)

This will cause all components to set their radiation fields using (if necessary) the properties of thisNode. Radiation objects can be queried using the following methods:

- **temperature(radiationTypes)** Returns the temperature (in Kelvin) of the radiation object. The optional radiationTypes array specifies which radiation types are to be queried.

- **flux(wavelength,radiationTypes)** Returns the flux (in ergs cm$^2$ s$^{-1}$ Hz$^{-1}$ ster$^{-1}$) of the radiation object at the given wavelength (specified in units of Å). The optional radiationTypes array specifies which radiation types are to be queried.
17.5.4. Mass Distributions

The `massDistribution` class, provided by the `Mass_Distributions` module, provides an object describing a distribution of mass in space, together with methods to query for the density, enclosed mass etc. of that mass distribution. Mass distribution objects make use of the `coordinates` subsystem (see §17.5.5) for specifying positions within mass distributions.

The base class provides the following methods:

- `symmetry` Returns one of the following labels to indicate the symmetry of the mass distribution:
  - `massDistributionSymmetryNone` Indicates that the mass distribution has no particular symmetry;
  - `massDistributionSymmetryCylindrical` Indicates that the mass distribution has cylindrical symmetry (conventionally around the \(z\)-axis);
  - `massDistributionSymmetrySpherical` Indicates that the mass distribution has spherical symmetry.

- `isDimensionless` Returns a Boolean indicating whether this is a dimensionless or dimensionful mass distribution;

- `density` Returns the density of the mass distribution at the supplied coordinates;

- `densityRadialMoment` Returns the \(n\)th moment of the integral of the density over radius, \(\int_0^\infty \rho(x)|x|^n \, dx\);

- `massEnclosedBySphere` Returns the mass enclosed by a sphere of given radius (centered on the origin);

- `potential` Returns the gravitational potential at the specified coordinates.

The `massDistributionSpherical` class extends the `massDistribution` base class with an additional method:

- `halfMassRadius` Returns the radius enclosing half of the mass of the density distribution.

Mass distributions are created using:

```
myMassDistribution => Mass_Distribution_Create(type)
```

where `type` is the name of the mass distribution (see below). After creation, the parameters of the profile must usually be initialized using:

```
call myMassDistribution%initialize(....)
```

Arguments for initialization vary for each mass distribution (see below).

Currently implemented mass distributions include:

- **nfw** An NFW [Navarro et al., 1997] density profile. Initialization is by

  ```
call myNfwProfile%initialize(scaleLength,concentration,densityNormalization,mass, &
& virialRadius,isDimensionless)
```

  All arguments are optional, but the combination given must be sufficient to allow the scale length and density normalization to be determined. The profile will be assumed to be dimensionful unless the `isDimensionless` argument specifies otherwise.

- **betaProfile** A \(\beta\)-profile, \(\rho(r) = \rho_0/[1 + (r/r_{core})^2]^{3\beta/2}\). Initialization is by
call myBetaProfile%initialize(beta,coreRadius,densityNormalization,mass, &
  outerRadius,isDimensionless)

beta = \beta and coreRadius = r_{core} must always be specified. The density normalization must be specified either by the densityNormalization argument, or by supplying both mass and outerRadius. The profile will be assumed to be dimensionful unless the isDimensionless argument specifies otherwise.

hernquist The Hernquist [Hernquist, 1990] profile. Initialization is by

call myHernquistProfile%initialize(scaleLength,densityNormalization,mass, &
  isDimensionless)

All arguments are optional, but the combination given must be sufficient to allow the scale length and density normalization to be determined unless the profile is dimensionless (in which case scale length and total mass are set to unity). The profile will be assumed to be dimensionful unless the isDimensionless argument specifies otherwise.

sersic The Sérsic [Sérsic, 1963] profile. Initialization is by

call mySersicProfile%initialize(index,halfMassRadius,densityNormalization,mass, &
  isDimensionless)

The Sérsic index must be specified. All other arguments are optional, but the combination given must be sufficient to allow the scale length and density normalization to be determined unless the profile is dimensionless (in which case scale length and total mass are set to unity). The profile will be assumed to be dimensionful unless the isDimensionless argument specifies otherwise.

17.5.5. Coordinates

The coordinate class, provided by the Coordinates module provides an object describing a position in three-dimensional space. Each extension of this class (currently, coordinateCartesian, coordinateCylindrical, and coordinateSpherical) supply methods to convert to and from Cartesian coordinates. The assignment operator (=) is overloaded such that coordinate objects of any class can be assigned to any other class and conversion to the appropriate coordinate system will happen automatically. A function accepting a class(coordinate) object can therefore convert it to, for example, spherical coordinates simply using

class(coordinate ), intent(in) :: coordinates
type (coordinateSpherical) :: coordinatesSpherical
coordinatesSpherical=coordinates

and thereby allow a position to be passed to it in any coordinate system.

Each extension of the base class also provides methods to get and set the values of each component of the relevant coordinate system (see §16.5.2 for complete details).
18. Auxilliary Methods

18.1. Conditional Stellar Mass Function

Empirical conditional stellar mass functions are used by GALACTICUS in calculations of halo mass function sampling. GALACTICUS implements the following calculations of tree processing times, which can be selected via the \([\text{conditionalStellarMassFunctionMethod}]\) input parameter.


Currently the only option, and selected using \([\text{conditionalStellarMassFunctionMethod}] = \text{Behroozi2010}\), this method adopts the fitting function of Behroozi et al. [2010]:

\[
\langle N_c(M_\star | M) \rangle \equiv \int_{M_\star}^{\infty} \phi_c(M'_\star) d \ln M'_\star = \frac{1}{2} \left[ 1 - \text{erf} \left( \frac{\log_{10} M_\star - \log_{10} f_{\text{SHMR}}(M)}{\sqrt{2} \sigma_{\log M_\star}} \right) \right]. \tag{18.1}
\]

Here, the function \(f_{\text{SHMR}}(M)\) is the solution of

\[
\log_{10} M = \log_{10} M_1 + \beta \log_{10} \left( \frac{M_\star}{M_{\star,0}} \right) + \frac{(M_\star/M_{\star,0})^\delta}{1 + (M_\star/M_{\star,0})^{-\gamma}} - 1/2. \tag{18.2}
\]

For satellites,

\[
\langle N_s(M_\star | M) \rangle \equiv \int_{M_\star}^{\infty} \phi_s(M'_\star) d \ln M'_\star = \langle N_c(M_\star | M) \rangle \frac{f_{\text{SHMR}}^{-1}(M_\star)}{M_{\text{sat}}} \alpha_{\text{sat}} \exp \left( -\frac{M_{\text{cut}}}{f_{\text{SHMR}}^{-1}(M_\star)} \right), \tag{18.3}
\]

where

\[
\frac{M_{\text{sat}}}{10^{12} M_\odot} = B_{\text{sat}} \left( \frac{f_{\text{SHMR}}^{-1}(M_\star)}{10^{12} M_\odot} \right)^{\beta_{\text{sat}}}, \tag{18.4}
\]

and

\[
\frac{M_{\text{cut}}}{10^{12} M_\odot} = B_{\text{cut}} \left( \frac{f_{\text{SHMR}}^{-1}(M_\star)}{10^{12} M_\odot} \right)^{\beta_{\text{cut}}}. \tag{18.5}
\]

By default, parameter values are taken from the fit of Leauthaud et al. [2011], specifically their \text{SIG\_MOD1} method for their \(z_1\) sample. These default values, and the GALACTICUS input parameters which can be used to adjust them are shown in Table 18.1. This method assumes that \(P_s(N|M_\star, M; \delta \ln M_\star)\) is a Poisson distribution while \(P_c(N|M_\star, M; \delta \ln M_\star)\) has a Bernoulli distribution, with each distribution’s free parameter fixed by requiring

\[
\phi(M_\star; M) \delta \ln M_\star = \sum_{N=0}^{\infty} NP(N|M_\star, M; \delta \ln M_\star) \tag{18.6}
\]
18. Auxiliary Methods

Table 18.1.: Parameters of the Behroozi et al. [2010] conditional stellar mass function model, along with their default values and the corresponding GALACTICUS input parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>GALACTICUS name</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1.0</td>
<td>[conditionalStellarMassFunctionBehrooziAlphaSatellite]</td>
</tr>
<tr>
<td>$\log_{10} M_1$</td>
<td>12.520</td>
<td>[conditionalStellarMassFunctionBehrooziLog10M1]</td>
</tr>
<tr>
<td>$\log_{10} M_{\star,0}$</td>
<td>10.916</td>
<td>[conditionalStellarMassFunctionBehrooziLog10Mstar0]</td>
</tr>
<tr>
<td>$\beta$</td>
<td>0.457</td>
<td>[conditionalStellarMassFunctionBehrooziBeta]</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.5666</td>
<td>[conditionalStellarMassFunctionBehrooziGamma]</td>
</tr>
<tr>
<td>$\delta$</td>
<td>1.53</td>
<td>[conditionalStellarMassFunctionBehrooziDelta]</td>
</tr>
<tr>
<td>$\sigma_{\log M_\star}$</td>
<td>0.206</td>
<td>[conditionalStellarMassFunctionBehrooziSigmaLogMstar]</td>
</tr>
<tr>
<td>$B_{\text{cut}}$</td>
<td>1.47</td>
<td>[conditionalStellarMassFunctionBehrooziBCut]</td>
</tr>
<tr>
<td>$B_{\text{sat}}$</td>
<td>10.62</td>
<td>[conditionalStellarMassFunctionBehrooziBSatellite]</td>
</tr>
<tr>
<td>$\beta_{\text{cut}}$</td>
<td>$-0.13$</td>
<td>[conditionalStellarMassFunctionBehrooziBetaCut]</td>
</tr>
<tr>
<td>$\beta_{\text{sat}}$</td>
<td>0.859</td>
<td>[conditionalStellarMassFunctionBehrooziBetaCut]</td>
</tr>
</tbody>
</table>

18.2. Tree Timing

Estimates of the time taken to process a merger tree are used in some halo sampling rate functions and may in future be used in load balancing algorithms. GALACTICUS implements the following calculations of tree processing times, which can be selected via the [timePerTreeMethod] input parameter.

18.2.1. File Method

Currently the only option, and selected using [timePerTreeMethod]=file, this method assumes that the time taken to process a tree is given by

$$\log_{10}[\tau_{\text{tree}}(M)] = \sum_{i=0}^{2} C_i (\log_{10} M)^i,$$

(18.7)

where $M$ is the root mass of the tree and the coefficients $C_i$ are read from a file, the name of which is specified via the [timePerTreeFitFileName] parameter. This file should be an XML document with the structure:

```xml
<timing>
  <fit>
    <coefficient>-0.73</coefficient>
    <coefficient>-0.20</coefficient>
    <coefficient>0.03</coefficient>
  </fit>
</timing>
```

where the array of coefficients give the values $C_0$, $C_1$ and $C_2$.

Note that, if GALACTICUS is run with [metaCollectTimingData]=true, then it will output measures of tree processing time to the output file (see §5.6.1). The analysis script scripts/analysis/treeTiming.pl can be used to extract tree timing data from such an output file and output fitting coefficients in the above format. It is used as follows:

```
treeTiming.pl <modelFile> [options.....]
```
where `<modelFile>` is the name of the GALACTICUS output file to analyze. The following options can be specified:

**accumulate** If present, this argument will cause new timing data from the `<modelFile>` to be accumulated with any timing data already present in the output file (which must be specified in this case). The fit is recomputed from the totality of the accumulated data;

**outputFile** If present, the timing data for individual trees together with the fitting coefficients will be output to the specified file;

**maxPoints** When accumulating trees to the output file, this parameter, if present, will limit the number of trees stored in the file to the given value. The oldest trees added to the file will be dropped first;

**plotFile** If present, a plot of the tree timing as a function of halo mass, together with the fitting function, will be output to the specified file.

Note that the output file will contain both the fitting coefficients in the format described above and, additionally, a list of tree root masses and processing times (necessary if you later want to append trees from another run to this file).
19. Source Code Documentation

file: `work/build/objects.nodes.components.Inc`
Description: Auto-generated file describing the hierarchy of node and component objects.
Code lines: N/A
Contained by: file `accretion.Bondi_Hoyle_Lyttleton.F90`
Used by: file `objects.nodes.F90`

19.1. Program units

file: `Bolshoi_Merger_Tree_File_Maker.F90`
Description: Contains a driver program for reading ASCII files of Rockstar merger trees from the Bolshoi Simulation and converting to GALACTICUS’s HDF5 format merger tree files.
Code lines: 53

program: `bolshoi_merger_tree_file_maker`
Description: Driver program for reading CSV files of Rockstar merger trees from the Bolshoi Simulation and converting to GALACTICUS’s HDF5 format merger tree files. Run scripts/aux/Bolshoi_Trees_Grab.pl on raw data files before converting to HDF5.
Code lines: 30
Contained by: file `Bolshoi_Merger_Tree_File_Maker.F90`
Modules used:
- command_arguments
- hdf5
- memory_management
- merger_tree_data_structure
- merger_trees_bolshoi

file: `Conditional_Mass_Function.F90`
Description: Contains a program which computes the conditional mass function in bins of mass for a fixed halo mass for use in calculation of constraints.
Code lines: 286

program: `conditional_mass_function`
Description: Computes the conditional mass function in bins of mass for a fixed halo mass for use in calculation of constraints.
Code lines: 265
Contained by: file `Conditional_Mass_Function.F90`
Modules used:
- command_arguments
- cosmology_functions
- fgs1
- galacticus_error
- geometry_surveys
- input_parameters
- io_hdf5
- iso_c_binding
- iso_varying_string
- memory_management
- numerical_integration
- numerical_ranges
function: mass_function_halo_mass_integrand
Description: Integral over halo mass function.
Code lines: 14
Contained by: program conditional_mass_function
Modules used: conditional_mass_functions halo_mass_function iso_c_binding

function: mass_function_time_integrand
Description: Integral over time.
Code lines: 15
Contained by: program conditional_mass_function
Modules used: iso_c_binding

function: mass_function_time_normalization_integrand
Description: Normalization integral over time.
Code lines: 10
Contained by: program conditional_mass_function
Modules used: iso_c_binding

file: Excursion_Sets.F90
Description: Contains a program which computes various quantities related to the excursion set formalism and stores them in an output file.
Code lines: 129

program: tests_excursion_sets
Description: Computes various quantities related to the excursion set formalism and stores them in an output file.
Code lines: 109
Contained by: file Excursion_Sets.F90
Modules used: cosmology_functions cosmology_parameters excursion_sets_barriers excursion_sets_first_crossings galacticus_display galacticus_error halo_mass_function input_parameters io_hdf5 iso_varying_string memory_management numerical_constants_math numerical_ranges power_spectra

file: Galacticus.F90
Description: GALACTICUS is a semi-analytic model of galaxy formation written by Andrew Benson <abenson@obs.carnegiescience.edu>.
Code lines: 59

program: galacticus
Description: The main GALACTICUS program.
Code lines: 38
Contained by: file Galacticus.F90
Modules used: galacticus_banner galacticus_error galacticus_hdf5 galacticus_tasks
19.1. Program units

file: Halo_Mass_Functions.F90
Description: Contains a code which computes dark matter halo mass functions and associated data.
Code lines: 62

program: halo_mass_functions
Description: Computes dark matter halo mass functions and associated data.
Code lines: 42
Contained by: file Halo_Mass_Functions.F90
Modules used: galacticus_error halo_mass_function_tasks
            input_parameters iso_varying_string
            memory_management

file: Mass_Function_Covariance.F90
Description: Compute covariance matrices for mass function estimates.
Code lines: 198

program: mass_function_covariance
Description: Compute covariance matrices for mass function estimates.
Code lines: 178
Contained by: file Mass_Function_Covariance.F90
Modules used: galacticus_error input_parameters
            io_hdf5 iso_varying_string
            memory_management numerical_constants_astronomical
            statistics_mass_function_covariance

file: Millennium_Merger_Tree_File_Maker.F90
Description: Contains a driver program for reading CSV files of merger trees from the Millennium Simulation database and converting to GALACTICUS’s HDF5 format merger tree files.
Code lines: 53

program: millennium_merger_tree_file_maker
Description: Driver program for reading CSV files of merger trees from the Millennium Simulation database and converting to GALACTICUS’s HDF5 format merger tree files.
Code lines: 32
Contained by: file Millennium_Merger_Tree_File_Maker.F90
Modules used: command_arguments hdf5
            memory_management merger_tree_data_structure
            merger_trees_millennium

file: Power_Spectra.F90
Description: Contains a code which computes power spectra and associated data.
Code lines: 62

program: power_spectra
Description: Computes power spectra and associated data.
Code lines: 42
19. Source Code Documentation

**Contained by:** file Power_Spectra.F90
**Modules used:**
- galacticus_error
- iso_varying_string
- memory_management
- power_spectrum_tasks

**file:** Simple_Merger_Tree_File_Maker.F90
**Description:** Contains a driver program for reading CSV files of simple merger trees and converting to GALACTICUS's HDF5 format merger tree files.
**Code lines:** 59

**program:** simple_merger_tree_file_maker
**Description:** Driver program for reading CSV files of simple merger trees and converting to GALACTICUS's HDF5 format merger tree files.
**Code lines:** 39
**Contained by:** file Simple_Merger_Tree_File_Maker.F90
**Modules used:**
- command_arguments
- hdf5
- input_parameters
- iso_varying_string
- memory_management
- merger_tree_data_structure
- merger_trees_simple

**file:** XRay_Absorption_ISM_Wilms2000.F90
**Description:** Contains a program which wraps the `dotbvabs` function (which implements the model of Wilms et al. 2000) from XSPEC to produce a table of X-ray absorption cross-sections in the ISM. This program assumes that various files from XSPEC have been downloaded into the aux/XSpec folder—usually this program will be run automatically as needed by the Galacticus::ISMCrossSections module.
**Code lines:** 164

**function:** fgabnd
**Description:** Function to return the abundance (relative to hydrogen) of elements. Required by `dotbvabs`.
**Code lines:** 28
**Contained by:** file XRay_Absorption_ISM_Wilms2000.F90
**Modules used:** galacticus_error

**subroutine:** xermsg
**Description:** Error message function required by `dotbvabs`.
**Code lines:** 12
**Contained by:** file XRay_Absorption_ISM_Wilms2000.F90
**Modules used:** galacticus_error

**program:** xray_absorption_ism_wilms2000
19.1. Program units

Description: Wraps the dotbvabs function (which implements the model of Wilms et al. 2000) from XSpec to produce a table of X-ray absorption cross-sections in the ISM. This program assumes that various files from XSpec have been downloaded into the aux/XSpec folder—usually this program will be run automatically as needed by the Galacticus::ISMColumnDensity module.

Code lines: 82

Contained by: file XRay_Absorption_ISM_Wilms2000.F90

Modules used:
- atomic_cross_sections_compton
- dates_and_times
- hdf5
- io_hdf5
- numerical_constants_prefixes
- numerical_constants_units
- numerical_ranges
subroutine: xwrite
  Description: Message display function required by dotbvabs.
  Code lines: 8
  Contained by: file XRay_Absorption_ISM_Wilms2000.F90

file: accretion.Bondi_Hoyle_Lyttleton.F90
  Description: Contains a module which implements calculations of Bondi-Hoyle-Lyttleton accretion (see Edgar 2004).
  Code lines: 67
module: bondi_hoyle_lyttleton_accretion
  Description: Implements calculations of Bondi-Hoyle-Lyttleton accretion (see Edgar 2004).
  Code lines: 47
  Contained by: file accretion.Bondi_Hoyle_Lyttleton.F90
  Used by: subroutine node_component_black_hole_standard_mass_accretion_rate
function: bondi_hoyle_lyttleton_accretion_radius
  Description: Computes the Bondi-Hoyle-Lyttleton accretion radius (in Mpc; Edgar 2004).
  Code lines: 14
  Contained by: module bondi_hoyle_lyttleton_accretion
  Modules used: ideal_gases_thermodynamics numerical_constants_physical
function: bondi_hoyle_lyttleton_accretion_rate
  Description: Computes the Bondi-Hoyle-Lyttleton accretion rate (in $M_{\odot}$ Gyr$^{-1}$; Edgar 2004).
  Code lines: 21
  Contained by: module bondi_hoyle_lyttleton_accretion
  Modules used: ideal_gases_thermodynamics numerical_constants_astronomical numerical_constants_physical

file: accretion.halo.F90
  Description: Contains a module which implements a class implementing accretion of gas from the IGM onto halos.
  Code lines: 1723
module: accretion_halos
Description: Implements a class implementing accretion of gas from the IGM onto halos. An implementation of null accretion from the IGM onto halos. An implementation of accretion from the IGM onto halos using simple truncation to mimic the effects of reionization. An implementation of accretion from the IGM onto halos using simple truncation to mimic the effects of reionization and accounting for cold mode accretion. An implementation of accretion from the IGM onto halos using filtering mass of the IGM calculated from an equation from Naoz and Barkana [2007].

Code lines: 1703

Contained by: file accretion.halo.F90

Modules used:
- abundances_structure
- chemical_abundances_structure
- galacticus_nodes
- iso_varying_string
- radiation_structure

Used by:
- subroutine count_properties
- subroutine establish_property_names
- subroutine galacticus_merger_tree_output
- subroutine node_component_hot_halo_cold_mode_rate_compute
- subroutine node_component_hot_halo_cold_mode_tree_initialize
- subroutine node_component_hot_halo_standard_rate_compute
subroutine node_component_hot_halo_standard_tree_initialize

subroutine: accretion_halos_hot_halo_output
  Description: Store hot halo properties in the GALACTICUS output file buffers.
  Code lines: 26
  Contained by: module accretion_halos
  Modules used: kind_numbers

subroutine: accretion_halos_hot_halo_output_count
  Description: Account for the number of hot halo cooling properties to be written to the GALACTICUS output file.
  Code lines: 13
  Contained by: module accretion_halos

subroutine: accretion_halos_hot_halo_output_names
  Description: Set names of hot halo properties to be written to the GALACTICUS output file.
  Code lines: 29
  Contained by: module accretion_halos
  Modules used: numerical_constants_astronomical

subroutine: accretion_halos_output_initialize
  Description: Initialize output in the halo accretion module.
  Code lines: 26
  Contained by: module accretion_halos
  Modules used: input_parameters

interface: accretionhalo
  Code lines: 3
  Contained by: module accretion_halos

function: accretionhaloaccretedmasschemicalsnull
  Description: Returns the mass of chemicals (in units of $M_{\odot}$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.
  Code lines: 10
  Contained by: module accretion_halos
  Modules used: galacticus_error

function: accretionhaloaccretedmassmetalsnull
  Description: Returns the mass of metals (in units of $M_{\odot}$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.
  Code lines: 10
  Contained by: module accretion_halos
  Modules used: galacticus_error

function: accretionhaloaccretedmassnull
  Description: Returns the mass (in units of $M_{\odot}$) of accreted from the IGM onto node in the given accretionMode. Used to initialize nodes.
  Code lines: 9
  Contained by: module accretion_halos
19.1. Program units

Modules used: galacticus_error

function: accretionhaloaccretionratechemicalsnull
Description: Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of chemicals from the IGM onto node in the given accretionMode.
Code lines: 10
Contained by: module accretion_halos
Modules used: galacticus_error

function: accretionhaloaccretionratemetalsnull
Description: Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of metals from the IGM onto node in the given accretionMode.
Code lines: 10
Contained by: module accretion_halos
Modules used: galacticus_error

function: accretionhaloaccretionratenull
Description: Returns the rate (in units of $M_\odot$ Gyr$^{-1}$) of accretion of mass from the IGM onto node in the given accretionMode.
Code lines: 9
Contained by: module accretion_halos
Modules used: galacticus_error

type: accretionhaloclass
Code lines: 70
Contained by: module accretion_halos

interface: accretionhalocoldmode
Description: Constructors for the coldMode halo accretion class.
Code lines: 4
Contained by: module accretion_halos

function: accretionhaloconstructordefault
Description: Return a pointer to the default accretionHalo object.
Code lines: 8
Contained by: module accretion_halos

function: accretionhaloconstructornamed
Description: Return a pointer to a newly created accretionHalo object of the specified type.
Code lines: 44
Contained by: module accretion_halos
Modules used: galacticus_error iso_varying_string

function: accretionhalofailedaccretedmassnull
Description: Returns the mass (in units of $M_\odot$) of that failed to accrete from the IGM onto node in the given accretionMode. Used to initialize nodes.
Code lines: 9
Contained by: module accretion_halos
Modules used: galacticus_error

function: accretionhalofailedaccretionratenull

889
19. Source Code Documentation

**Description:** Returns the rate (in units of $M_\odot\,\text{Gyr}^{-1}$) of failed accretion of mass from the IGM onto node in the given accretionMode.

**Code lines:** 9
**Contained by:** module accretion_halos
**Modules used:** galacticus_error

**subroutine:** accretionhaloinitialize

**Description:** Initialize the default accretionHalo object.

**Code lines:** 61
**Contained by:** module accretion_halos
**Modules used:** galacticus_error, input_parameters, iso_varying_string

**function:** accretionhaloisfinalizable

**Description:** Return true if this object can be finalized.

**Code lines:** 7
**Contained by:** module accretion_halos

**interface:** accretionhalonaozbarkana2007

**Description:** Constructors for the naozBarkana2007 halo accretion class.

**Code lines:** 4
**Contained by:** module accretion_halos

**interface:** accretionhalonull

**Description:** Constructors for the null halo accretion class.

**Code lines:** 3
**Contained by:** module accretion_halos

**interface:** accretionhalosimple

**Description:** Constructors for the simple halo accretion class.

**Code lines:** 4
**Contained by:** module accretion_halos

**interface:** assignment(=)

**Code lines:** 2
**Contained by:** module accretion_halos

**function:** coldmodeaccretedmass

**Description:** Computes the mass of baryons accreted into node.

**Code lines:** 10
**Contained by:** module accretion_halos
**Modules used:** galacticus_nodes

**function:** coldmodeaccretedmasschemicals

**Description:** Computes the mass of chemicals accreted (in $M_\odot$) onto node from the intergalactic medium.

**Code lines:** 20
**Contained by:** module accretion_halos
**Modules used:** chemical_abundances_structure, galacticus_nodes
function: coldmodeaccretedmassmetals
Description: Computes the mass of abundances accreted (in $M_\odot$) onto node from the intergalactic medium.
Code lines: 11
Contained by: module accretion_halos
Modules used: galacticus_nodes

function: coldmodeaccretionrate
Description: Computes the baryonic accretion rate onto node.
Code lines: 10
Contained by: module accretion_halos
Modules used: galacticus_nodes

function: coldmodeaccretionratechemicals
Description: Computes the rate of mass of chemicals accretion (in $M_\odot$/Gyr) onto node from the intergalactic medium. Assumes a primordial mixture of hydrogen and helium and that accreted material is in collisional ionization equilibrium at the virial temperature.
Code lines: 22
Contained by: module accretion_halos
Modules used: chemical_abundances_structure galacticus_nodes

function: coldmodeaccretionratemetals
Description: Computes the rate of mass of abundance accretion (in $M_\odot$/Gyr) onto node from the intergalactic medium.
Code lines: 11
Contained by: module accretion_halos
Modules used: galacticus_nodes

function: coldmodechemicalmasses
Description: Compute the masses of chemicals accreted (in $M_\odot$) onto node from the intergalactic medium.
Code lines: 57
Contained by: module accretion_halos
Modules used: chemical_abundances_structure chemical_reaction_rates_utilities chemical_states cosmology_parameters dark_matter_halo_scales galacticus_nodes intergalactic_medium_state numerical_constants_astronomical numerical_constants_atomic numerical_constants_math numerical_constants_physical numerical_constants_prefixes shocks_1d

function: coldmodecoldmodefraction
Description: Computes the fraction of accretion occuring in the specified mode.
Code lines: 69
Contained by: module accretion_halos
Modules used: chemical_abundances_structure chemical_reaction_rates_utilities chemical_states cooling_functions cosmology_parameters dark_matter_halo_scales galacticus_nodes numerical_constants_atomic numerical_constants_math numerical_constants_astronomical numerical_constants_prefixes
function: coldmodeconstructor
Description: Default constructor for the coldMode halo accretion class.
Code lines: 11
Contained by: module accretion_halos

function: coldmodedefaultconstructor
Description: Default constructor for the coldMode halo accretion class.
Code lines: 45
Contained by: module accretion_halos
Modules used: cosmology_functions galacticus_error
input_parameters intergalactic_medium_state

function: coldmodefailedaccretedmass
Description: Computes the mass of baryons accreted into node.
Code lines: 10
Contained by: module accretion_halos
Modules used: galacticus_nodes

function: coldmodefailedaccretionrate
Description: Computes the baryonic accretion rate onto node.
Code lines: 10
Contained by: module accretion_halos
Modules used: galacticus_nodes

subroutine: coldmodefromsimple
Description: Assign a simple halo accretion object to a coldMode halo accretion object.
Code lines: 12
Contained by: module accretion_halos

function: naozbarkana2007constructor
Description: Default constructor for the naozBarkana2007 halo accretion class.
Code lines: 9
Contained by: module accretion_halos

function: naozbarkana2007defaultconstructor
Description: Default constructor for the naozBarkana2007 halo accretion class.
Code lines: 7
Contained by: module accretion_halos

function: naozbarkana2007failedfraction
Description: Returns the velocity scale to use for node. Use the virial velocity.
Code lines: 26
Contained by: module accretion_halos
Modules used: cosmology_parameters galacticus_error
 galacticus_nodes intergalactic_medium_state

subroutine: naozbarkana2007fromsimple
Description: Assign a simple halo accretion object to a naozBarkana2007 halo accretion object.
Code lines: 12
19.1. Program units

**Contained by:** module accretion_halos

**function:** nullaccretedmass

*Description:* Computes the mass of baryons accreted into node.

*Code lines:* 10

*Contained by:* module accretion_halos

*Modules used:* galacticus_nodes

**function:** nullaccretedmasschemicals

*Description:* Computes the mass of chemicals accreted (in $M_\odot$) onto node from the intergalactic medium.

*Code lines:* 12

*Contained by:* module accretion_halos

*Modules used:* chemical_abundances_structure galacticus_nodes

**function:** nullaccretedmassmetals

*Description:* Computes the mass of abundances accreted (in $M_\odot$) onto node from the intergalactic medium.

*Code lines:* 11

*Contained by:* module accretion_halos

*Modules used:* galacticus_nodes

**function:** nullaccretionrate

*Description:* Computes the baryonic accretion rate onto node.

*Code lines:* 10

*Contained by:* module accretion_halos

*Modules used:* galacticus_nodes

**function:** nullaccretionratechemicals

*Description:* Computes the rate of mass of chemicals accretion (in $M_\odot$/Gyr) onto node from the intergalactic medium.

*Code lines:* 12

*Contained by:* module accretion_halos

*Modules used:* chemical_abundances_structure galacticus_nodes

**function:** nullaccretionratemetals

*Description:* Computes the rate of mass of abundance accretion (in $M_\odot$/Gyr) onto node from the intergalactic medium.

*Code lines:* 11

*Contained by:* module accretion_halos

*Modules used:* galacticus_nodes

**function:** nullconstructor

*Description:* Default constructor for the null halo accretion class.

*Code lines:* 7

*Contained by:* module accretion_halos

*Modules used:* input_parameters

**function:** nullfailedaccretedmass

*Description:* Computes the mass of baryons accreted into node.

*Code lines:* 10
19. Source Code Documentation

**Contained by:** module accretion_halos

**Modules used:** galacticus_nodes

**function: nullfailedaccretionrate**

**Description:** Computes the baryonic accretion rate onto node.

**Code lines:** 10

**Contained by:** module accretion_halos

**Modules used:** galacticus_nodes

**function: simpleaccretedmass**

**Description:** Computes the mass of baryons accreted into node.

**Code lines:** 22

**Contained by:** module accretion_halos

**Modules used:** cosmology_parameters galacticus_nodes

**function: simpleaccretedmasschemicals**

**Description:** Computes the mass of chemicals accreted (in $M_\odot$) onto node from the intergalactic medium.

**Code lines:** 24

**Contained by:** module accretion_halos

**Modules used:** chemical_abundances_structure galacticus_nodes

**function: simpleaccretedmassmetals**

**Description:** Computes the mass of abundances accreted (in $M_\odot$) onto node from the intergalactic medium.

**Code lines:** 12

**Contained by:** module accretion_halos

**Modules used:** galacticus_nodes

**function: simpleaccretionrate**

**Description:** Computes the baryonic accretion rate onto node.

**Code lines:** 38

**Contained by:** module accretion_halos

**Modules used:** cosmology_parameters galacticus_nodes

**function: simpleaccretionratechemicals**

**Description:** Computes the rate of mass of chemicals accretion (in $M_\odot$/Gyr) onto node from the intergalactic medium. Assumes a primordial mixture of hydrogen and helium and that accreted material is in collisional ionization equilibrium at the virial temperature.

**Code lines:** 26

**Contained by:** module accretion_halos

**Modules used:** chemical_abundances_structure galacticus_nodes

**function: simpleaccretionratemetals**

**Description:** Computes the rate of mass of abundance accretion (in $M_\odot$/Gyr) onto node from the intergalactic medium.

**Code lines:** 12

**Contained by:** module accretion_halos

**Modules used:** galacticus_nodes

**function: simpleaccretionratetotal**
19.1. Program units

**Description:** Returns the total accretion rate from the IGM onto a halo (including dark matter).

**Code lines:** 11

**Contained by:** module accretion_halos

**Modules used:** galacticus_nodes

**function: simplechemicalmasses**

**Description:** Compute the masses of chemicals accreted (in $M_\odot$) onto node from the intergalactic medium.

**Code lines:** 39

**Contained by:** module accretion_halos

**Modules used:** chemical_abundances_structure, chemical_reaction_rates_utilities, chemical_states, cosmology_parameters, dark_matter_halo_scales, galacticus_nodes, numerical_constants_astronomical

**function: simpleconstructor**

**Description:** Default constructor for the simple halo accretion class.

**Code lines:** 32

**Contained by:** module accretion_halos

**Modules used:** atomic_data, chemical_abundances_structure, galacticus_error, galacticus_nodes, input_parameters

**function: simpledefaultconstructor**

**Description:** Default constructor for the simple halo accretion class.

**Code lines:** 85

**Contained by:** module accretion_halos

**Modules used:** cosmology_functions, galacticus_error, input_parameters, intergalactic_medium_state

**function: simplefailedaccretedmass**

**Description:** Computes the mass of baryons accreted into node.

**Code lines:** 23

**Contained by:** module accretion_halos

**Modules used:** cosmology_parameters, dark_matter_halo_scales, galacticus_nodes

**function: simplefailedaccretionrate**

**Description:** Computes the baryonic accretion rate onto node.

**Code lines:** 37

**Contained by:** module accretion_halos

**Modules used:** cosmology_parameters, dark_matter_halo_scales, galacticus_nodes

**function: simplefailedfraction**

**Description:** Returns the fraction of potential accretion onto a halo from the IGM which fails.

**Code lines:** 15

**Contained by:** module accretion_halos

**Modules used:** galacticus_nodes
19. Source Code Documentation

function: simplemasstotal
Description: Returns the total node mass.
Code lines: 11
Contained by: module accretion_halos
Modules used: galacticus_nodes

function: simplevelocityscale
Description: Returns the velocity scale to use for node. Use the virial velocity.
Code lines: 12
Contained by: module accretion_halos
Modules used: dark_matter_halo_scales galacticus_nodes

Description: An implementation of accretion from the IGM onto halos using filtering mass of the IGM calculated from an equation from Naoz and Barkana [2007].
Code lines: 106

interface: accretionhalonaozbarkana2007
Description: Constructors for the naozBarkana2007 halo accretion class.
Code lines: 4

interface: assignment(=)
Code lines: 2

function: naozbarkana2007constructor
Description: Default constructor for the naozBarkana2007 halo accretion class.
Code lines: 9

function: naozbarkana2007defaultconstructor
Description: Default constructor for the naozBarkana2007 halo accretion class.
Code lines: 7

function: naozbarkana2007failedfraction
Description: Returns the velocity scale to use for node. Use the virial velocity.
Code lines: 26
Modules used: cosmology_parameters galacticus_error galacticus_nodes intergalactic_medium_state

subroutine: naozbarkana2007fromsimple
Description: Assign a simple halo accretion object to a naozBarkana2007 halo accretion object.
Code lines: 12
19.1. Program units

file: accretion.halo.cold_mode.F90
Description: An implementation of accretion from the IGM onto halos using simple truncation to mimic the effects of reionization and accounting for cold mode accretion.
Code lines: 397

interface: accretionhalocoldmode
Description: Constructors for the coldMode halo accretion class.
Code lines: 4
Contained by: file accretion.halo.cold_mode.F90

interface: assignment(=)
Code lines: 2
Contained by: file accretion.halo.cold_mode.F90

function: coldmodeaccretedmass
Description: Computes the mass of baryons accreted into node.
Code lines: 10
Contained by: file accretion.halo.cold_mode.F90
Modules used: galacticus_nodes

function: coldmodeaccretedmasschemicals
Description: Computes the mass of chemicals accreted (in \( M_\odot \)) onto node from the intergalactic medium.
Code lines: 20
Contained by: file accretion.halo.cold_mode.F90
Modules used: chemical_abundances_structure galacticus_nodes

function: coldmodeaccretedmassmetals
Description: Computes the mass of abundances accreted (in \( M_\odot \)) onto node from the intergalactic medium.
Code lines: 11
Contained by: file accretion.halo.cold_mode.F90
Modules used: galacticus_nodes

function: coldmodeaccretionrate
Description: Computes the baryonic accretion rate onto node.
Code lines: 10
Contained by: file accretion.halo.cold_mode.F90
Modules used: galacticus_nodes

function: coldmodeaccretionratechemicals
Description: Computes the rate of mass of chemicals accretion (in \( M_\odot/\text{Gyr} \)) onto node from the intergalactic medium. Assumes a primordial mixture of hydrogen and helium and that accreted material is in collisional ionization equilibrium at the virial temperature.
Code lines: 22
Contained by: file accretion.halo.cold_mode.F90
Modules used: chemical_abundances_structure galacticus_nodes

function: coldmodeaccretionratemetals
Description: Computes the rate of mass of abundance accretion (in \( M_\odot/\text{Gyr} \)) onto node from the intergalactic medium.
function: coldmodechemicalmasses
Description: Compute the masses of chemicals accreted (in $M_\odot$) onto node from the intergalactic medium.

function: coldmodecoldmodefraction
Description: Computes the fraction of accretion occurring in the specified mode.

function: coldmodeconstructor
Description: Default constructor for the coldMode halo accretion class.

function: coldmodedefaultconstructor
Description: Default constructor for the coldMode halo accretion class.

function: coldmodefailedaccretedmass
Description: Computes the mass of baryons accreted into node.

function: coldmodefailedaccretionrate
Description: Computes the baryonic accretion rate onto node.
subroutine: coldmodefromsimple
Description: Assign a simple halo accretion object to a coldMode halo accretion object.
Code lines: 12
Contained by: file accretion.halo.cold_mode.F90

file: accretion.halo.null.F90
Description: An implementation of null accretion from the IGM onto halos.
Code lines: 154

interface: accretionhalonull
Description: Constructors for the null halo accretion class.
Code lines: 3
Contained by: file accretion.halo.null.F90

function: nullaccretedmass
Description: Computes the mass of baryons accreted into node.
Code lines: 10
Contained by: file accretion.halo.null.F90
Modules used: galacticus_nodes

function: nullaccretedmasschemicals
Description: Computes the mass of chemicals accreted (in $M_\odot$) onto node from the intergalactic medium.
Code lines: 12
Contained by: file accretion.halo.null.F90
Modules used: galacticus_nodes, chemical_abundances_structure

function: nullaccretedmassmetals
Description: Computes the mass of abundances accreted (in $M_\odot$) onto node from the intergalactic medium.
Code lines: 11
Contained by: file accretion.halo.null.F90
Modules used: galacticus_nodes

function: nullaccretionrate
Description: Computes the baryonic accretion rate onto node.
Code lines: 10
Contained by: file accretion.halo.null.F90
Modules used: galacticus_nodes

function: nullaccretionratechemicals
Description: Computes the rate of mass of chemicals accretion (in $M_\odot$/Gyr) onto node from the intergalactic medium.
Code lines: 12
Contained by: file accretion.halo.null.F90
Modules used: chemical_abundances_structure, galacticus_nodes

function: nullaccretionratemetals
Description: Computes the rate of mass of abundance accretion (in $M_\odot$/Gyr) onto node from the intergalactic medium.
Code lines: 11
19. Source Code Documentation

**Contained by:** file `accretion.halo.null.F90`
**Modules used:** `galacticus_nodes`

**function: nullconstructor**
**Description:** Default constructor for the null halo accretion class.
**Code lines:** 7
**Contained by:** file `accretion.halo.null.F90`
**Modules used:** `galacticus_nodes`

**function: nullfailedaccretedmass**
**Description:** Computes the mass of baryons accreted into node.
**Code lines:** 10
**Contained by:** file `accretion.halo.null.F90`
**Modules used:** `input_parameters`

**function: nullfailedaccretionrate**
**Description:** Computes the baryonic accretion rate onto node.
**Code lines:** 10
**Contained by:** file `accretion.halo.null.F90`
**Modules used:** `galacticus_nodes`

**file: accretion.halo.simple.F90**
**Description:** An implementation of accretion from the IGM onto halos using simple truncation to mimic the effects of reionization.
**Code lines:** 526
**Modules used:** `radiation_structure`

**interface: accretionhalosimple**
**Description:** Constructors for the simple halo accretion class.
**Code lines:** 4
**Contained by:** file `accretion.halo.simple.F90`

**function: simpleaccretedmass**
**Description:** Computes the mass of baryons accreted into node.
**Code lines:** 22
**Contained by:** file `accretion.halo.simple.F90`
**Modules used:** `cosmology_parameters` `galacticus_nodes`

**function: simpleaccretedmasschemicals**
**Description:** Computes the mass of chemicals accreted (in $M_\odot$) onto node from the intergalactic medium.
**Code lines:** 24
**Contained by:** file `accretion.halo.simple.F90`
**Modules used:** `chemical_abundances_structure` `galacticus_nodes`

**function: simpleaccretedmassmetals**
**Description:** Computes the mass of abundances accreted (in $M_\odot$) onto node from the intergalactic medium.
**Code lines:** 12
**Contained by:** file `accretion.halo.simple.F90`
**Modules used:** `galacticus_nodes`
function: simpleaccretionrate
Description: Computes the baryonic accretion rate onto node.
Code lines: 38
Contained by: file accretion.halo.simple.F90
Modules used: cosmology_parameters galacticus_nodes

function: simpleaccretionratechemicals
Description: Computes the rate of mass of chemicals accretion (in $M_\odot$/Gyr) onto node from the intergalactic medium. Assumes a primordial mixture of hydrogen and helium and that accreted material is in collisional ionization equilibrium at the virial temperature.
Code lines: 26
Contained by: file accretion.halo.simple.F90
Modules used: chemical_abundances_structure galacticus_nodes

function: simpleaccretionratemetals
Description: Computes the rate of mass of abundance accretion (in $M_\odot$/Gyr) onto node from the intergalactic medium.
Code lines: 12
Contained by: file accretion.halo.simple.F90
Modules used: galacticus_nodes

function: simpleaccretionratetotal
Description: Returns the total accretion rate from the IGM onto a halo (including dark matter).
Code lines: 11
Contained by: file accretion.halo.simple.F90
Modules used: galacticus_nodes

function: simplechemicalmasses
Description: Compute the masses of chemicals accreted (in $M_\odot$) onto node from the intergalactic medium.
Code lines: 39
Contained by: file accretion.halo.simple.F90
Modules used: chemical_abundances_structure chemical_reaction_rates_utilities chemical_states cosmology_parameters dark_matter_halo_scales galacticus_nodes numerical_constants_astronomical

function: simpleconstructor
Description: Default constructor for the simple halo accretion class.
Code lines: 32
Contained by: file accretion.halo.simple.F90
Modules used: atomic_data chemical_abundances_structure galacticus_error galacticus_nodes input_parameters

function: simpledefaultconstructor
Description: Default constructor for the simple halo accretion class.
Code lines: 85
Contained by: file accretion.halo.simple.F90
19. Source Code Documentation

**Modules used:**
- cosmology_functions
- galacticus_error
- input_parameters
- intergalactic_medium_state

**function: simplefailedaccretedmass**
*Description:* Computes the mass of baryons accreted into node.
*Code lines:* 23
*Contained by:* file accretion.halo.simple.F90
*Modules used:* cosmology_parameters
dark_matter_halo_scales
galacticus_nodes

**function: simplefailedaccretionrate**
*Description:* Computes the baryonic accretion rate onto node.
*Code lines:* 37
*Contained by:* file accretion.halo.simple.F90
*Modules used:* cosmology_parameters
dark_matter_halo_scales
galacticus_nodes

**function: simplefailedfraction**
*Description:* Returns the fraction of potential accretion onto a halo from the IGM which fails.
*Code lines:* 15
*Contained by:* file accretion.halo.simple.F90
*Modules used:* galacticus_nodes

**function: simplesmasstotal**
*Description:* Returns the total node mass.
*Code lines:* 11
*Contained by:* file accretion.halo.simple.F90
*Modules used:* galacticus_nodes

**function: simplevelocityscale**
*Description:* Returns the velocity scale to use for node. Use the virial velocity.
*Code lines:* 12
*Contained by:* file accretion.halo.simple.F90
*Modules used:* dark_matter_halo_scales
galacticus_nodes

**file: accretion_disks.ADAF.F90**
*Description:* Contains a module which implements calculations of properties of ADAFs based on the implementation of Benson and Babul [2009].
*Code lines:* 718

**module: accretion_disks_adaf**
*Description:* Implements calculations of properties of ADAFs based on the implementation of Benson and Babul [2009].
*Code lines:* 698
*Contained by:* file accretion_disks.ADAF.F90
*Modules used:* iso_varying_string
tables
*Used by:* subroutine accretion_disks_initialize
module accretion_disks_switched

**subroutine: accretion_disk_adaf_tabulate**
19.1. Program units

**Description:** Tabulate jet power and spin-up efficiency for an ADAF.

**Code lines:** 45

**Contained by:** module **accretion_disks_adaf**

**Modules used:**
- black_hole_fundamentals
- numerical_constants_physical

**function: accretion_disk_jet_power_adaf**

**Description:** Computes the jet power for an ADAF in units of $M_\odot \ (\text{km/s})^2 \ \text{Gyr}^{-1}$.

**Code lines:** 25

**Contained by:** module **accretion_disks_adaf**

**Modules used:**
- galacticus_nodes

**function: accretion_disk_radiative_efficiency_adaf**

**Description:** Computes the radiative efficiency for an ADAF.

**Code lines:** 18

**Contained by:** module **accretion_disks_adaf**

**Modules used:**
- accretion_disks_shakura_sunyaev
- galacticus_nodes

**subroutine: accretion_disks_adaf_get_parameters**

**Description:** Initialize the module by reading in parameter values.

**Code lines:** 147

**Contained by:** module **accretion_disks_adaf**

**Modules used:**
- galacticus_error
- input_parameters

**subroutine: accretion_disks_adaf_initialize**

**Description:** Test if this method is to be used and set procedure pointer appropriately.

**Code lines:** 15

**Contained by:** module **accretion_disks_adaf**

**function: adaf_alpha**

**Description:** Returns the effective value of $\alpha$ for an ADAF.

**Code lines:** 28

**Contained by:** module **accretion_disks_adaf**

**function: adaf_angular_momentum**

**Description:** Returns the specific angular momentum of accreted material in the ADAF.

**Code lines:** 15

**Contained by:** module **accretion_disks_adaf**

**function: adaf_bh_jet_power**

**Description:** Returns the power of the black hole-launched jet from an ADAF.

**Code lines:** 28

**Contained by:** module **accretion_disks_adaf**

**Modules used:**
- black_hole_fundamentals

**function: adaf_disk_jet_power**

**Description:** Returns the power of the disk-launched jet from an ADAF.

**Code lines:** 23

**Contained by:** module **accretion_disks_adaf**

**Modules used:**
- black_hole_fundamentals
function: adaf_disk_jet_power_from_black_hole
Description: Returns the power extracted from the black hole by the disk-launched jet from an ADAF.
Code lines: 15
Contained by: module accretion_disks_adaf

function: adaf_enthalpy
Description: Returns the relativistic enthalpy of the ADAF.
Code lines: 15
Contained by: module accretion_disks_adaf

function: adaf_enthalpy-angular_momentum_product
Description: Return the product of enthalpy and angular momentum for the ADAF.
Code lines: 30
Contained by: module accretion_disks_adaf
Modules used: black_hole_fundamentals

function: adaf_field_enhancement
Description: Returns the field enhancement factor, \( g \), in the ADAF.
Code lines: 26
Contained by: module accretion_disks_adaf
Modules used: black_hole_fundamentals

function: adaf_fluid-angular_velocity
Description: Returns the angular velocity of the rotating fluid with respect to the local inertial observer (ZAMO).
Code lines: 16
Contained by: module accretion_disks_adaf
Modules used: black_hole_fundamentals

function: adaf_gamma
Description: Returns the net relativistic boost factor from the fluid frame of an ADAF to an observer at rest at infinity. The input quantities are in natural units.
Code lines: 16
Contained by: module accretion_disks_adaf

function: adaf_gamma_phi
Description: Returns the \( \phi \) component relativistic boost factor from the fluid frame of an ADAF to an observer at rest at infinity. The input quantities are in natural units.
Code lines: 17
Contained by: module accretion_disks_adaf
Modules used: black_hole_fundamentals

function: adaf_gamma_r
Description: Returns the \( r \) component relativistic boost factor from the fluid frame of an ADAF to an observer at rest at infinity. The input quantities are in natural units.
Code lines: 16
Contained by: module accretion_disks_adaf

function: adaf_height
Description: Return the (dimensionless) height in an ADAF at given radius, for a black hole of given blackHoleSpin and a flow with viscosity parameter adafViscosityAlpha.

Code lines: 21
Contained by: module accretion_disks_adaf
Modules used: black_hole_fundamentals

function: adaf_temperature

Description: Return the (dimensionless) velocity in an ADAF at given radius, for a black hole of given blackHoleSpin and a flow with viscosity parameter adafViscosityAlpha.

Code lines: 28
Contained by: module accretion_disks_adaf
Modules used: black_hole_fundamentals

function: adaf_v

Description: Computes the spin up rate of the black hole in thisBlackHole due to accretion from an ADAF disk.

Code lines: 31
Contained by: module accretion_disks_adaf
Modules used: black_hole_fundamentals

file: accretion_disks.Eddington_limited.F90

Description: Contains a module which implements calculations of properties of accretion disks which ignore the details of accretion physics and assume that black hole jets have a power that is a fixed fraction of the Eddington luminosity.

Code lines: 112

module: accretion_disks_eddington

Description: Implements calculations of properties of accretion disks which ignore the details of accretion physics and assume that black hole jets have a power that is a fixed fraction of the Eddington luminosity.

Code lines: 91
Contained by: file accretion_disks.Eddington_limited.F90
Used by: subroutine accretion_disks_initialize

function: accretion_disk_jet_power_eddington

Description: Computes the jet power for an Eddington-limited accretion disk.

Code lines: 11
Contained by: module accretion_disks_eddington
Modules used: black_hole_fundamentals galacticus_nodes numerical_constants_physical

function: accretion_disk_radiative_efficiency_eddington

Description: Computes the radiative efficiency for an Eddington-limited accretion disk.

Code lines: 9
19. Source Code Documentation

**module** accretion_disks_eddington

**included by:** module galacticus_nodes

**subroutine:** accretion_disks_eddington_initialize

**description:** Test if this method is to be used and set procedure pointer appropriately.

**code lines:** 38

**contained by:** module accretion_disks_eddington

**modules used:** galacticus_nodes

**function:** black_hole_spin_up_rate_eddington

**description:** Computes the spin-up rate of the black hole in thisNode due to accretion from an Eddington-limited accretion disk. This is always zero, as no physical model is specified for this accretion disk method.

**code lines:** 10

**contained by:** module accretion_disks_eddington

**modules used:** galacticus_nodes

**file:** accretion_disks.F90

**description:** Contains a module which implements calculations related to accretion disks.

**code lines:** 149

**module:** accretion_disks

**description:** Implements calculations related to accretion disks.

**code lines:** 129

**contained by:** file accretion_disks.F90

**modules used:** iso_varying_string

**used by:** subroutine node_component_black_hole_standard_mass_accretion_rate subroutine node_component_black_hole_standard_output subroutine node_component_black_hole_standard_output_properties subroutine node_component_black_hole_standard_radiative_efficiency subroutine node_component_black_hole_standard_rate_compute

**function:** accretion_disk_jet_power

**description:** Computes the jet power for an accretion disk in units of $M_\odot\ (\text{km/s})^2\ \text{Gyr}^{-1}$.

**code lines:** 14

**contained by:** module accretion_disks

**modules used:** galacticus_nodes

**function:** accretion_disk_radiative_efficiency

**description:** Computes the radiative efficiency for an accretion disk.

**code lines:** 14

**contained by:** module accretion_disks

**modules used:** galacticus_nodes

**subroutine:** accretion_disks_initialize

**description:** Initialize the accretion disk module.

**code lines:** 58

**contained by:** module accretion_disks
19.1. Program units

Modules used: accretion_disks_adaf accretion_disks_eddington
accretion_disks_shakura_sunyaev accretion_disks_switched
galacticus_error input_parameters

function: black_hole_spin_up_rate
Description: Computes the spin up rate of the black hole in thisNode due to accretion from an accretion disk.
Code lines: 15
Contained by: module accretion_disks
Modules used: galacticus_nodes

file: accretion_disks.Shakura_Sunyaev.F90
Description: Contains a module which implements calculations of properties of thin Shakura-Sunyaev accretion disks.
Code lines: 115

module: accretion_disks_shakura_sunyaev
Description: Implements calculations of properties of thin Shakura-Sunyaev accretion disks.
Code lines: 95
Contained by: file accretion_disks.Shakura_Sunyaev.F90
Used by: function accretion_disk_radiative_efficiency_shakura_sunyaev subroutine accretion_disks_initialize
module accretion_disks_switched

function: accretion_disk_jet_power_shakura_sunyaev
Description: Computes the jet power for a Shakura-Sunyaev (thin) accretion disk, using the expressions from Meier (2001; his equations 4 and 5).
Code lines: 38
Contained by: module accretion_disks_shakura_sunyaev
Modules used: black_hole_fundamentals galacticus_nodes
numerical_constants_astronomical

function: accretion_disk_radiative_efficiency_shakura_sunyaev
Description: Computes the radiative efficiency for a Shakura-Sunyaev (thin) accretion disk.
Code lines: 10
Contained by: module accretion_disks_shakura_sunyaev
Modules used: black_hole_fundamentals galacticus_nodes

subroutine: accretion_disks_shakura_sunyaev_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 15
Contained by: module accretion_disks_shakura_sunyaev
Modules used: iso_varying_string

function: black_hole_spin_up_rate_shakura_sunyaev
Description: Computes the spin up rate of the black hole in thisBlackHole due to accretion from a Shakura-Sunyaev (thin) accretion disk.
Code lines: 13
Contained by: module accretion_disks_shakura_sunyaev
Modules used: black_hole_fundamentals galacticus_nodes
file: accretion_disks.spectra.F90

Description: Contains a module that implements calculations of accretion disk spectra.

Code lines: 453

module: accretion_disk_spectra

Description: Implements calculations of accretion disk spectra. An implementation of the accretion disk spectra class for tabulated spectra read from file. An implementation of the accretion disk spectra class for tabulated spectra read from file.

Code lines: 433

Contained by: file accretion_disks.spectra.F90

Modules used:fgsl      
iso_varying_string

Used by:
function radiation_intergalactic_-
background_internal_update

interface: accretiondiskspectra

Code lines: 3

Contained by: module accretion_disk_spectra

type: accretiondiskspectra\nclass

Code lines: 21

Contained by: module accretion_disk_spectra

function: accretiondiskspectra\nclass\nconstructor\ndefault

Description: Return a pointer to the default accretionDiskSpectra object.

Code lines: 8

Contained by: module accretion_disk_spectra

function: accretiondiskspectra\nclass\nconstructor\nnamed

Description: Return a pointer to a newly created accretionDiskSpectra object of the specified type.

Code lines: 30

Contained by: module accretion_disk_spectra

 Modules used:galacticus_error
iso_varying_string

interface: accretiondiskspectra\nclass\nfile

Description: Constructors for the file accretion disk spectra class.

Code lines: 4

Contained by: module accretion_disk_spectra

interface: accretiondiskspectra\nclass\nhopkins2007

Description: Constructors for the hopkins2007 accretion disk spectra class.

Code lines: 3

Contained by: module accretion_disk_spectra

subroutine: accretiondiskspectra\nclass\ninitialize

Description: Initialize the default accretionDiskSpectra object.

Code lines: 47

Contained by: module accretion_disk_spectra
19.1. Program units

Modules used:  
galacticus_error  
iso_varying_string  
input_parameters

function: accretion_disk_spectra.isFinalizable
Description: Return true if this object can be finalized.
Code lines: 7
Contained by: module accretion_disk_spectra

function: accretion_disk_spectra.spectrumNull
Description: Returns the spectrum (in units of $L_\odot \text{ Hz}^{-1}$) of the accretion disk at the given wavelength (in units of Å) for node.
Code lines: 9
Contained by: module accretion_disk_spectra
Modules used: galacticus_error

function: fileConstructor
Description: Constructor for the file accretion disk spectra class.
Code lines: 16
Contained by: module accretion_disk_spectra
Modules used: array_utilities galacticus_error

function: fileDefaultConstructor
Description: Default constructor for the file accretion disk spectra class.
Code lines: 26
Contained by: module accretion_disk_spectra
Modules used: input_parameters

subroutine: fileDestructor
Description: Default destructor for the file accretion disk spectra class.
Code lines: 13
Contained by: module accretion_disk_spectra
Modules used: memory_management numerical_interpolation

subroutine: fileLoadFile
Description: Load a file of AGN spectra.
Code lines: 22
Contained by: module accretion_disk_spectra
Modules used: galacticus_error io_hdf5

function: fileSpectrum
Description: Return the accretion disk spectrum for tabulated spectra.
Code lines: 38
Contained by: module accretion_disk_spectra
Modules used: iso_c_binding
numerical_constants_astronomical
numerical_constants_physical numerical_interpolation

function: hopkins2007Constructor
Description: Constructor for the hopkins2007 accretion disk spectra class.
Code lines: 18
19. Source Code Documentation

```
Contained by: module accretion_disk_spectra
Modules used: galacticus_input_paths string_handling system_command

Description: An implementation of the accretion disk spectra class for tabulated spectra read from file.
Code lines: 55

Description: Constructors for the Hopkins2007 accretion disk spectra class.
Code lines: 3

function: Hopkins2007 constructor
Description: Constructor for the Hopkins2007 accretion disk spectra class.
Code lines: 18
Modules used: galacticus_input_paths string_handling system_command

file: accretion_disks.spectra.file.F90
Description: An implementation of the accretion disk spectra class for tabulated spectra read from file.
Code lines: 189
Modules used: fgsl

interface: accretion_disks.spectra.file
Description: Constructors for the file accretion disk spectra class.
Code lines: 4
Contained by: file accretion_disks.spectra.file.F90

function: file constructor
Description: Constructor for the file accretion disk spectra class.
Code lines: 16
Contained by: file accretion_disks.spectra.file.F90
Modules used: array_utilities galacticus_error

function: file default constructor
Description: Default constructor for the file accretion disk spectra class.
Code lines: 26
Contained by: file accretion_disks.spectra.file.F90
Modules used: input_parameters

subroutine: file destructor
Description: Default destructor for the file accretion disk spectra class.
Code lines: 13
Contained by: file accretion_disks.spectra.file.F90
Modules used: memory_management numerical_interpolation

subroutine: file load file
```
19.1. Program units

Description: Load a file of AGN spectra.
Code lines: 22
Contained by: file accretion_disks.spectra.file.F90
Modules used: galacticus_error io_hdf5

function: filespectrum
Description: Return the accretion disk spectrum for tabulated spectra.
Code lines: 38
Contained by: file accretion_disks.spectra.file.F90
Modules used: iso_c_binding numerical_constants_astronomical numerical_constants_physical numerical_interpolation

file: accretion_disks.switched.F90
Description: Contains a module which implements calculations of properties of accretion disks which switch between thin and ADAF depending on the accretion rate.
Code lines: 247

module: accretion_disks_switched
Description: Implements calculations of properties of accretion disks which switch between thin and ADAF depending on the accretion rate.
Code lines: 226
Contained by: file accretion_disks.switched.F90
Modules used: accretion_disks_adaf accretion_disks_shakura_sunyaev
Used by: subroutine accretion_disks_initialize

function: accretion_disk_jet_power_switched
Description: Computes the jet power for a switching accretion disk.
Code lines: 11
Contained by: module accretion_disks_switched
Modules used: galacticus_nodes

function: accretion_disk_radiative_efficiency_switched
Description: Computes the radiative efficiency for a switching accretion disk.
Code lines: 14
Contained by: module accretion_disks_switched
Modules used: galacticus_nodes

function: accretion_disk_switched_adaf_fraction
Description: Decide which type of accretion disk to use.
Code lines: 40
Contained by: module accretion_disks_switched
Modules used: black_hole_fundamentals galacticus_nodes

function: accretion_disk_switched_adaf_radiative_efficiency_scaling
Description: Determine the scaling of radiative efficiency of the ADAF component in a switched accretion disk.
Code lines: 34
Contained by: module accretion_disks_switched
Modules used: black_hole_fundamentals galacticus_nodes
**subroutine:** accretion_disks_switched_initialize  
*Description:* Test if this method is to be used and set procedure pointer appropriately.  
*Code lines:* 75  
*Contained by:* module accretion_disks_switched  
*Modules used:*  
  - input_parameters  
  - iso_varying_string

**function:** black_hole_spin_up_rate_switched  
*Description:* Computes the spin up rate of the black hole in *thisBlackHole* due to accretion from a switching accretion disk.  
*Code lines:* 14  
*Contained by:* module accretion_disks_switched  
*Modules used:* galacticus_nodes

**file:** atomic.cross_sections.Compton.F90  
*Description:* Contains a module which implements calculation of the Compton cross-section.  
*Code lines:* 43  

**module:** atomic_cross_sections_compton  
*Description:* Implements calculation of the Compton cross-section.  
*Code lines:* 23  
*Contained by:* file atomic.cross_sections.Compton.F90  
*Used by:* program xray_absorption_ism_wilms2000

**function:** atomic_cross_section_compton  
*Description:* Returns the Compton cross section (in cm$^2$) for the specified photonEnergy (in keV) from Klein and Nishina [1929].  
*Code lines:* 13  
*Contained by:* module atomic_cross_sections_compton  
*Modules used:*  
  - numerical_constants_physical  
  - numerical_constants_prefixes  
  - numerical_constants_units

**file:** atomic.cross_sections.ionization.photo.F90  
*Description:* Contains a module that implements calculations of atomic photo-ionization cross-sections.  
*Code lines:* 115  

**module:** atomic_cross_sections_ionization_photo  
*Description:* Implements calculations of atomic photo-ionization cross-sections.  
*Code lines:* 95  
*Contained by:* file atomic.cross_sections.ionization.photo.F90  
*Used by:*  
  - function cross_section_h_gamma_to_hplus_electron  
  - function intergalactic_medium_state_internal_odes  
  - subroutine radiation_intergalactic_background_internal_initialize

**function:** atomic_cross_section_ionization_photo  
*Description:* Return the cross-section (in units of cm$^2$) for a given atom in a given ionization state at the specified wavelength (given in units of Å).  
*Code lines:* 14
19.1. Program units

**Contained by:** module `atomic_cross_sections_ionization_photo`

**subroutine:** `atomic_cross_section_ionization_photo_initialize`

*Description:* Initialize the atomic photo ionization cross-section module.

*Code lines:* 53

**Contained by:** module `atomic_cross_sections_ionization_photo`

**Modules used:**
- `atomic_cross_sections_ionization_photo_verner`
- `galacticus_error`
- `input_parameters`

**file:** `atomic.cross_sections.ionization.photo.Verner.F90`

*Description:* Contains a module which computes cross sections for photo-ionization for all ionization stages of all atoms from H to Zn.

*Code lines:* 2032

**module:** `atomic_cross_sections_ionization_photo_verner`

*Description:* Computes cross sections for photo-ionization for all ionization stages of all atoms from H to Zn ($Z = 30$) by use of the following fit parameters:

- Outer shells of the Opacity Project (OP) elements: Verner et al. [1996]
- Inner shells of all elements, and outer shells of the non-OP elements: Verner and Yakovlev [1995]

Original version (`phfit2.f`) written by D. A. Verner (Version 2. March 25, 1996). Inner-shell ionization energies of some low-ionized species are slightly improved to fit smoothly the experimental inner-shell ionization energies of neutral atoms.

*Code lines:* 2012

**Contained by:** file `atomic.cross_sections.ionization.photo.Verner.F90`

**Used by:**
- subroutine `atomic_cross_section_ionization_photo_initialize`

**function:** `atomic_cross_section_ionization_photo_verner`

*Description:* Computes the cross section for photo-ionization (in units of cm$^2$) at the specified wavelength for all ionization stages of all atoms from H to Zn ($Z = 30$) by use of the following fit parameters:

- Outer shells of the Opacity Project (OP) elements: Verner et al. [1996]
- Inner shells of all elements, and outer shells of the non-OP elements: Verner and Yakovlev [1995]

Original version (`phfit2.f`) written by D. A. Verner (Version 2. March 25, 1996). Inner-shell ionization energies of some low-ionized species are slightly improved to fit smoothly the experimental inner-shell ionization energies of neutral atoms.

*Code lines:* 83

**Contained by:** module `atomic_cross_sections_ionization_photo_verner`

**Modules used:**
- `numerical_constants_physical`
- `numerical_constants_units`

**subroutine:** `atomic_cross_section_ionization_photo_verner_initialize`

*Description:* Initializes the “Verner” atomic photo-ionization cross section module.
19. Source Code Documentation

Code lines: 11
Contained by: module atomic_cross_sections_ionization_photo_verner
Modules used: iso_varying_string

file: atomic.data.F90
Description: Contains a module which provides various atomic data.
Code lines: 347

module: atomic_data
Description: Provides various atomic data.
Code lines: 327
Contained by: file atomic.data.F90
Used by: function simpleconstructor
subroutine abundances_initialize
subroutine abundances_set_metallicity
subroutine stellar_astrophysics_file_initialize
subroutine supernovae_type_ia_nagashima_initialization
subroutine stellar_population_properties_instantaneous_initialize

function: abundance_pattern_lookup
Description: Returns the position in the atoms() array of an element specified by atomic number, name or short label.
Code lines: 33
Contained by: module atomic_data
Modules used: galacticus_error string_handling

function: atom_lookup
Description: Returns the position in the atoms() array of an element specified by atomic number, name or short label.
Code lines: 43
Contained by: module atomic_data
Modules used: galacticus_error string_handling

function: atomic_abundance
Description: Returns the abundance by mass of a given atom in a given abundance pattern.
Code lines: 32
Contained by: module atomic_data

function: atomic_data_atoms_count
Description: Return the number of atomic species known in this module.
Code lines: 10
Contained by: module atomic_data

subroutine: atomic_data_initialize
Description: Ensure that the module is initialized by reading in data.
Code lines: 117
Contained by: module atomic_data
Modules used: fox_dom
                    galacticus_error
                    galacticus_input_paths io_xml
19.1. Program units

**iso_varying_string**  
string_handling

**function: atomic_mass**  
*Description:* Returns the atomic mass of an element specified by atomic number, name or short label.  
*Code lines:* 21  
*Contained by:* module atomic_data

**function: atomic_short_label**  
*Description:* Return the short label for an atom.  
*Code lines:* 19  
*Contained by:* module atomic_data

**type: atomicdata**  
*Description:* Data type for storing atomic data.  
*Code lines:* 7  
*Contained by:* module atomic_data

**file: atomic.ionization_potentials.F90**  
*Description:* Contains a module that implements calculations of atomic ionization potentials.  
*Code lines:* 106

**module: atomic.ionization_potentials**  
*Description:* Implements calculations of atomic photo-ionization cross-sections.  
*Code lines:* 84  
*Contained by:* file atomic.ionization_potentials.F90  
*Modules used:* iso_varying_string  
*Used by:* function gaunt_factor_sutherland1998  
function intergalactic_medium_state INTERNAL_ODES

**function: atomic.ionization_potential**  
*Description:* Return the ionization potential (in units of eV) for a given atom in a given ionization state.  
*Code lines:* 10  
*Contained by:* module atomic.ionization_potentials

**subroutine: atomic.ionization_potential_initialize**  
*Description:* Initialize the atomic ionization potential module.  
*Code lines:* 52  
*Contained by:* module atomic.ionization_potentials  
*Modules used:* atomic.ionization_potentials_verner  
galacticus_error  
input_parameters

**file: atomic.ionization_potentials.Verner.F90**  
*Description:* Contains a module which computes ionization potentials for all ionization stages of all atoms from H to Zn using data taken from Dima Verner’s code.  
*Code lines:* 536

**module: atomic.ionization_potentials_verner**  
*Description:* Computes ionization potentials for all ionization stages of all atoms from H to Zn using data taken from Dima Verner’s code.
19. Source Code Documentation

**Code lines:** 513  
**Contained by:** file `atomic.ionization_potentials.Verner.F90`  
**Used by:** subroutine `atomic_ionization_potential_initialize`

**function:** `atomic_ionization_potential_verner`  
**Description:** Return the ionization potential (in units of electron volts) for the ion with given `atomicNumber` and `electronNumber` using data taken from Dima Verner’s code.  
**Code lines:** 8  
**Contained by:** module `atomic_ionization_potentials_verner`

**subroutine:** `atomic_ionization_potentials_verner_initialize`  
**Description:** Initializes the “Verner” atomic ionization potential module.  
**Code lines:** 486  
**Contained by:** module `atomic_ionization_potentials_verner`  
**Modules used:** `iso_varying_string`

**file:** `atomic.radiation.gaunt_factors.F90`  
**Description:** Contains a module that implements calculations of Gaunt factors for Bremsstrahlung emission from ions.  
**Code lines:** 111  
**Module:** `atomic_radiation.gaunt_factors`  
**Description:** Implements calculations of Gaunt factors for Bremsstrahlung emission from ions.  
**Code lines:** 91  
**Contained by:** file `atomic.radiation.gaunt_factors.F90`  
**Modules used:** `iso_varying_string`  
**Used by:** function `intergalactic_medium_state_internal_odes`

**function:** `gaunt_factor`  
**Description:** Return the Gaunt factor for the given ion and temperature.  
**Code lines:** 11  
**Contained by:** module `atomic_radiation.gaunt_factors`

**subroutine:** `gaunt_factor_initialize`  
**Description:** Initialize the Gaunt factor module.  
**Code lines:** 52  
**Contained by:** module `atomic_radiation.gaunt_factors`  
**Modules used:** `galacticus_error`  
`gaunt_factors_sutherland1998`  
`input_parameters`

**file:** `atomic.radiation.gaunt_factors.Sutherland1998.F90`  
**Description:** Contains a module which computes Gaunt factors for Bremsstrahlung emission from ions using the fitting functions of Sutherland [1998].  
**Code lines:** 85  
**Module:** `gaunt_factors_sutherland1998`  
**Description:** Contains a function which computes Gaunt factors for Bremsstrahlung emission from ions using the fitting functions of Sutherland [1998].
19.1. Program units

function: gaunt_factor_sutherland1998
Description: Compute energy averaged Gaunt factors for thermal electron distributions using the tabulations and fits of Sutherland [1998].

subroutine: gaunt_factor_sutherland1998_initialize
Description: Initializes the “Sutherland1998” Gaunt Factor module.

file: atomic.rates.excitation.collisional.F90
Description: Contains a module that implements calculations of collisional excitation rates.

module: atomic_rates_excitation_collisional
Description: Implements calculations of collisional excitation rates.

function: collisional_excitation_cooling_rate
Description: Return the collisional excitation cooling rate, in units of J/m^3/s, for ion of given atomicNumber and electronNumber at temperature T (in Kelvin).

subroutine: collisional_excitation_rate_initialize
Description: Initialize the collisional excitation rate module.

file: atomic.rates.excitation.collisional.ScholzWalters91.F90
Description: Contains a module which computes cooling rates due to collisional excitation using the fitting functions of Scholz and Walters [1991].
module: atomic_rates_excitation_collisional_scholzwalters91
Description: Contains a function which computes cooling rates due to collisional excitation using the fitting functions of Scholz and Walters [1991].
Code lines: 69
Contained by: file atomic.rates.excitation.collisional.ScholzWalters91.F90
Used by: subroutine collisional_excitation_cooling_rate

function: collisional_excitation_cooling_rate
Description: Return collisional excitation cooling rates, in units of J/m^3/s, for ion Ion at temperature T (in Kelvin) using the fitting functions of Scholz and Walters [1991].
Code lines: 44
Contained by: module atomic_rates_excitation_collisional_scholzwalters91
Modules used: galacticus_error

subroutine: collisional_excitation_cooling_rate_scholzwalters91_initialize
Description: Initializes the "ScholzWalters91" collisional excitation rate module.
Code lines: 10
Contained by: module atomic_rates_excitation_collisional_scholzwalters91
Modules used: iso_varying_string

file: atomic.rates.ionization.collisional.F90
Description: Contains a module that implements calculations of atomic collisional ionization rates.
Code lines: 113

module: atomic_rates_ionization_collisional
Description: Implements calculations of atomic collisional ionization rates.
Code lines: 93
Contained by: file atomic.rates.ionization.collisional.F90
Modules used: iso_varying_string
Used by: subroutine chemical_hydrogen_rate_h__electron_to_hplus_2electron
function intergalactic_medium_state__internal_odes

function: atomic_rate_ionization_collisional
Code lines: 12
Contained by: module atomic_rates_ionization_collisional

subroutine: atomic_rate_ionization_collisional_initialize
Description: Initialize the atomic collisional ionization rate module.
Code lines: 53
Contained by: module atomic_rates_ionization_collisional
Modules used: atomic_rates_ionization_collisional__verner

file: atomic.rates.ionization.collisional.Verner.F90
Description: Contains a module which computes rates of atomic direct collisional ionization using the fitting function of Voronov (1997; Version 2, March 24, 1997). Based on the code originally written by Dima Verner.
Code lines: 497
module: atomic_rates_ionization_collisional_verner
Description: Computes rates of atomic direct collisional ionization using the fitting function of Voronov (1997; Version 2, March 24, 1997). Based on the code originally written by Dima Verner.
Code lines: 475
Contained by: file atomic.rates.ionization.collisional.Verner.F90
Used by: subroutine atomic_rate_ionization_collisional_initialize

function: atomic_rate_ionization_collisional_verner
Description: Computes the rate coefficient of direct collisional ionization (in units of cm$^3$s$^{-1}$) at the specified temperature for all ions of atoms with $Z < 28$ by use of the fits from Voronov (1997; Version 2, March 24, 1997). Based on the code originally written by Dima Verner. The ionization state passed to this function should be that of the atom/ion prior to ionization.
Code lines: 35
Contained by: module atomic_rates_ionization_collisional_verner
Modules used: numerical_constants_physical numerical_constants_units

subroutine: atomic_rate_ionization_collisional_verner_initialize
Description: Initializes the “Verner” atomic collisional ionization rate module.
Code lines: 11
Contained by: module atomic_rates_ionization_collisional_verner
Modules used: iso_varying_string

file: atomic.rates.recombination.dielectronic.Arnaud85.F90
Description: Contains a module which computes rates of dielectronic recombination.
Code lines: 246

module: atomic_rates_recombination_dielectronic_arnaud1985
Description: Contains a function which computes rates of dielectronic recombination.
Code lines: 224
Contained by: file atomic.rates.recombination.dielectronic.Arnaud85.F90
Used by: subroutine dielectronic_recombination_rate_arnaud1985_initialize

subroutine: dielectronic_recom_arnaud1985_initialize
Description: Initializes the “Arnaud1985” dielectronic recombination rate module.
Code lines: 10
Contained by: module atomic_rates_recombination_dielectronic_arnaud1985
Modules used: iso_varying_string

function: dielectronic_recombination_rate_arnaud1985
Description: This function calculates rates of dielectric recombination for all ionization stages of all elements from H to Ni ($Z = 28$) by use of the fits from Aldrovandi and Pequignot [1973], Shull and van Steenberg [1982] and Arnaud and Rothenflug [1985]. Input parameters: 
- atomicNumber: atomic number;
- electronNumber: number of electrons;
- temperature: temperature [K].

Output parameter: rate coefficient [cm$^3$ s$^{-1}$].

Code lines: 22

Contained by: module atomic_rates_recombination_dielectronic_arnaud1985

Modules used: galacticus_error

subroutine: set_coefficients
19.1. Program units

Description: Populate the array of fitting function coefficients for the Arnaud1985 dielectronic recombination rates.

Code lines: 172

Contained by: module atomic_rates_recombination_dielectronic_arnaud1985

file: atomic.rates.recombination.dielectronic.F90
Description: Contains a module that implements calculations of dielectronic recombination rates.
Code lines: 114

module: atomic_rates_recombination_dielectronic
Description: Implements calculations of dielectronic recombination rates.
Code lines: 92

Contained by: file atomic.rates.recombination.dielectronic.F90
Modules used: iso_varying_string
Used by: function intergalactic_medium_state_internal_odes

function: dielectronic_recombination_rate
Description: Return the dielectronic recombination rate (in units of cm$^3$ s$^{-1}$) for the ion of given atomicNumber and electronNumber at the given temperature (in Kelvin).
Code lines: 12

Contained by: module atomic_rates_recombination_dielectronic

subroutine: dielectronic_recombination_rate_initialize
Description: Initialize the dielectronic recombination rate module.
Code lines: 52

Contained by: module atomic_rates_recombination_dielectronic
Modules used: atomic_rates_recombination_dielectronic_arnaud1985 galacticus_error

file: atomic.rates.recombination.radiative.F90
Description: Contains a module that implements calculations of atomic radiative recombination rates.
Code lines: 108

module: atomic_rates_recombination_radiative
Description: Implements calculations of atomic radiative recombination rates.
Code lines: 88

Contained by: file atomic.rates.recombination.radiative.F90
Modules used: iso_varying_string
Used by: subroutine chemical_hydrogen_rate hplus_electron_to_h_photon function intergalactic_medium_state_internal_odes

function: atomic_rate_recombination_radiative
Code lines: 13

Contained by: module atomic_rates_recombination_radiative

subroutine: atomic_rate_recombination_radiative_initialize
Description: Initialize the atomic radiative recombination rate module.
Code lines: 53
19. Source Code Documentation

**Contained by:** module `atomic_rates_recombination_radiative`  
**Modules used:** `atomic_rates_recombination_radiative_verner`  
`galacticus_error`  
`input_parameters`  

**file:** `atomic.rates.recombination.radiative.Verner.F90`  
**Description:** Contains a module which computes rates of atomic radiative recombination using a compilation of fitting functions. Based on the code originally written by Dima Verner.  
**Code lines:** 655

**module:** `atomic_rates_recombination_radiative_verner`  
**Description:** Computes rates of atomic radiative recombination using a compilation of fitting functions. Based on the code originally written by Dima Verner.  
**Code lines:** 634  
**Contained by:** file `atomic.rates.recombination.radiative.Verner.F90`  
**Used by:** subroutine `atomic_rate_recombination_radiative_initialize`  

**function:** `atomic_rate_recombination_radiative_verner`  
**Description:** Computes the rate coefficient of radiative recombination (in units of cm$^3$ s$^{-1}$) at the specified temperature for all ions of all elements from H through Zn (selected by the atomicNumber and the ionizationState of the recombined ion) use of the following fits:  
- H-like, He-like, Li-like, Na-like: [Verner and Ferland, 1996];  
- Other ions of C, N, O, Ne: [Pequignot et al., 1991], refitted by Verner & Ferland formula to ensure correct asymptotes;  
- Fe XVII-XXIII: [Arnaud and Raymond, 1992];  
- Fe I-XV: refitted by Verner & Ferland formula to ensure correct asymptotes;  
- Other ions of Mg, Si, S, Ar, Ca, Fe, Ni: [Shull and van Steenberg, 1982];  
- Other ions of Na, Al: [Landini and Fossi, 1990];  
- Other ions of F, P, Cl, K, Ti, Cr, Mn, Co (excluding Ti I-II, Cr I-IV, Mn I-V, Co I): [Landini and Fossi, 1991];  
- All other species: interpolations of the power-law fits.  

Based on the code originally written by Dima Verner. The ionization state passed to this function should be that of the atom/ion post recombination.  
**Code lines:** 108  
**Contained by:** module `atomic_rates_recombination_radiative_verner`  
**Modules used:** `atomic_rates_recombination_radiative_data`  
`galacticus_error`  

**subroutine:** `atomic_rate_recombination_radiative_verner_initialize`  
**Description:** Initializes the “Verner” atomic collisional ionization rate module.  
**Code lines:** 11  
**Contained by:** module `atomic_rates_recombination_radiative_verner`
19.1. Program units

**Modules used:** iso_varying_string

**file:** atomic.rates.recombination.radiative.data.F90
- **Description:** Contains a module that provides options for use in calculations of atomic radiative recombination rates.
- **Code lines:** 31

**module:** atomic_rates_recombination_radiative_data
- **Description:** Provides options used in calculations of atomic radiative recombination rates.
- **Code lines:** 9
- **Contained by:** file atomic.rates.recombination.radiative.data.F90
- **Used by:** function atomic_rate_recombination_radiative_verner, function intergalactic_medium_state_internal_odes

**file:** bivar.F90
- **Code lines:** 2983

**module:** bivar
- **Code lines:** 2982
- **Contained by:** file bivar.F90
- **Used by:** function interpolate_2d_irregular_array

**subroutine:** idbvip
- **Code lines:** 281
- **Contained by:** module bivar

**subroutine:** idcldp
- **Code lines:** 225
- **Contained by:** module bivar

**subroutine:** idgrid
- **Code lines:** 383
- **Contained by:** module bivar

**subroutine:** idlctn
- **Code lines:** 349
- **Contained by:** module bivar
- **Modules used:** omp_lib

**subroutine:** idpdrv
- **Code lines:** 245
- **Contained by:** module bivar

**subroutine:** idptip
- **Code lines:** 470
- **Contained by:** module bivar

**subroutine:** idsfft
- **Code lines:** 288
- **Contained by:** module bivar
19. Source Code Documentation

subroutine: idtang
Code lines: 589
Contained by: module bivar

function: idxchg
Code lines: 132
Contained by: module bivar

file: black_holes.binaries.initial_radius.F90
Description: Contains a module which implements calculations of black hole binary initial separations.
Code lines: 102

module: black_hole_binary_initial_radii
Description: Implements calculations of black hole binary initial separations.
Code lines: 82
Contained by: file black_holes.binaries.initial_radius.F90
Modules used: iso_varying_string
Used by: subroutine node_component_black_hole_standard_satellite_merging

function: black_hole_binary_initial_radius
Description: Computes the initial radius of a newly formed black hole binary.
Code lines: 62
Contained by: module black_hole_binary_initial_radii
Modules used: black_hole_binary_initial_radii_spheroid_size black_hole_binary_initial_radii_tidal_radius
black_hole_binary_initial_radii_volonteri_2003 galacticus_error
volonteri_2003 galacticus_nodes input_parameters

file: black_holes.binaries.initial_radius.Volonteri_2003.F90
Description: Contains a module which implements a black hole binary initial separation which follows that of Volonteri et al. [2003].
Code lines: 65

module: black_hole_binary_initial_radii_volonteri_2003
Description: Implements a black hole binary initial separation which follows that of Volonteri et al. [2003].
Code lines: 43
Contained by: file black_holes.binaries.initial_radius.Volonteri_2003.F90
Used by: function black_hole_binary_initial_radius

subroutine: black_hole_binary_initial_radii_volonteri_2003_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 9
Contained by: module black_hole_binary_initial_radii_volonteri_2003
Modules used: iso_varying_string
function: black_hole_binary_initial_radius_volonteri_2003
  Description: Returns an initial separation for binary black holes using the method of Volonteri et al. [2003], with the assumption that the local velocity dispersion is approximately the dark matter halo virial velocity.
  Code lines: 19
  Contained by: module black_hole_binary_initial_radii_volonteri_2003
  Modules used: dark_matter_halo_scales galacticus_nodes numerical_constants_physical

file: black_holes.binaries.initial_radius.spheroid_size_fraction.F90
  Description: Contains a module which implements a black hole binary initial separation which is a fixed fraction of the scale radius of the larger of the host and satellite spheroids.
  Code lines: 77

module: black_hole_binary_initial_radii_spheroid_size
  Description: Implements a black hole binary initial separation which is a fixed fraction of the scale radius of the larger of the host and satellite spheroids.
  Code lines: 56
  Contained by: file black_holes.binaries.initial_radius.spheroid_size_fraction.F90
  Used by: function black_hole_binary_initial_radius

subroutine: black_hole_binary_initial_radii_spheroid_size_initialize
  Description: Test if this method is to be used and set procedure pointer appropriately.
  Code lines: 23
  Contained by: module black_hole_binary_initial_radii_spheroid_size
  Modules used: input_parameters iso_varying_string

function: black_hole_binary_initial_radius_spheroid_size
  Description: Returns a initial separation for a binary black holes that is a fixed fraction of the scale radius of the larger of the host and satellite spheroids.
  Code lines: 14
  Contained by: module black_hole_binary_initial_radii_spheroid_size
  Modules used: galacticus_nodes

file: black_holes.binaries.initial_radius.tidal_radius.F90
  Description: Contains a module which implements a black hole binary initial separation based on tidal disruption of the satellite galaxy.
  Code lines: 95

module: black_hole_binary_initial_radii_tidal_radius
  Description: Implements a black hole binary initial separation based on tidal disruption of the satellite galaxy.
  Code lines: 73
  Contained by: file black_holes.binaries.initial_radius.tidal_radius.F90
  Modules used: galacticus_nodes
  Used by: function black_hole_binary_initial_radius
subroutine: black_hole_binary_initial_radii_tidal_radius_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 9
Contained by: module black_hole_binary_initial_radii_tidal_radius
Modules used: iso_varying_string

function: black_hole_binary_initial_radius_tidal_radius
Description: Returns an initial separation for a binary black holes through tidal disruption.
Code lines: 32
Contained by: module black_hole_binary_initial_radii_tidal_radius
Modules used: galactic_structure_enclosed_masses galactic_structure_options root_finder

function: tidal_radius_root
Description: Root function used in solving for the radius of tidal disruption of a satellite galaxy.
Code lines: 10
Contained by: module black_hole_binary_initial_radii_tidal_radius
Modules used: galactic_structure_enclosed_masses galactic_structure_options

Description: Contains a module which implements a black hole binary recoil velocity which follows the formulae in Campanelli et al. [2007], derived from post-Newtonian evaluations.
Code lines: 137

module: black_hole_binary_recoil_velocities_standard
Description: Implements a black hole binary recoil velocity which follows the formulae in Campanelli et al. [2007], derived from post-Newtonian evaluations.
Code lines: 114
Contained by: file black_holes.binaries.recoil_velocity.Campanelli2007.F90
Modules used: fgsl
Used by: function black_hole_binary_recoil_velocity_standard
subroutine galacticus_state_retrieve
velocity
subroutine galacticus_state_snapshot
subroutine galacticus_state_store

function: black_hole_binary_recoil_velocity_standard
Description: Returns the recoil velocity of a black hole binary, accounting for the parallel and perpendicular velocities, plus that of the binary’s center of mass. Constants used are retrieved from the articles by: Koppitz et al. [2007] for \( H = (7.3 \pm 0.3) \times 10^4 \) km/s, Gonzalez et al. [2007b] for \( A = 1.2 \times 10^4 \) km/s \( B = -0.93 \), Gonzalez et al. [2007a] for \( K \cos(\delta \theta) = (6, -5.3) \times 10^4 \) km/s and \( K = (6.0 \pm 0.1) \times 10^4 \) km/s.
Code lines: 42
Contained by: module black_hole_binary_recoil_velocities_standard
Modules used: numerical_constants_math pseudo_random

subroutine: black_hole_binary_recoil_velocity_standard_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 9
Contained by: module black_hole_binary_recoil_velocities_standard
19.1. Program units

**Modules used:** iso_varying_string

**subroutine:** black_hole_binary_recoil_velocity_standard_snapshot
- **Description:** Store a snapshot of the random number generator internal state.
- **Code lines:** 7
- **Contained by:** module black_hole_binary_recoil_velocities_standard

**subroutine:** black_hole_binary_recoil_velocity_standard_state_retrieve
- **Description:** Write the stored snapshot of the random number state to file.
- **Code lines:** 10
- **Contained by:** module black_hole_binary_recoil_velocities_standard

**subroutine:** black_hole_binary_recoil_velocity_standard_state_store
- **Description:** Write the stored snapshot of the random number state to file.
- **Code lines:** 10
- **Contained by:** module black_hole_binary_recoil_velocities_standard

**Modules used:** pseudo_random

**file:** black_holes.binaries.recoil_velocity.F90
- **Description:** Contains a module which implements calculations of black hole binary recoil velocities.
- **Code lines:** 101

**module:** black_hole_binary_recoil_velocities
- **Description:** Implements calculations of black hole binary recoil velocities.
- **Code lines:** 79
- **Contained by:** file black_holes.binaries.recoil_velocity.F90
- **Modules used:** iso_varying_string
- **Used by:** subroutine node_component_black_hole_binary_recoil_velocity_standard_merge_black_holes subroutine node_component_black_hole_binary_recoil_velocity_standard_satellite_merging

**function:** black_hole_binary_recoil_velocity
- **Description:** Computes the recoil velocity of a black hole binary.
- **Code lines:** 59
- **Contained by:** module black_hole_binary_recoil_velocities
- **Modules used:** black_hole_binary_recoil_velocities_standard black_hole_recoil_velocities_null standard galacticus_error input_parameters

**file:** black_holes.binaries.recoil_velocity.null.F90
- **Description:** Contains a module which implements a zero black hole binary recoil velocity.
- **Code lines:** 54

**module:** black_hole_recoil_velocities_null
- **Description:** Implements a zero black hole binary recoil velocity.
- **Code lines:** 31
- **Contained by:** file black_holes.binaries.recoil_velocity.null.F90
- **Used by:** function black_hole_binary_recoil_velocity
function: black_hole_binary_recoil_velocity_null
Description: Returns a zero recoil velocity for black hole binary mergers.
Code lines: 7
Contained by: module black_hole_recoil_velocities_null

subroutine: black_hole_binary_recoil_velocity_null_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 9
Contained by: module black_hole_recoil_velocities_null
Modules used: iso_varying_string

file: black_holes.binaries.separation_growth_rate.F90
Description: Contains a module which implements calculations of black hole binary separation growth rate.
Code lines: 102

module: black_hole_binary_separations
Description: Implements calculations of black hole binary separation growth rate.
Code lines: 80
Contained by: file black_holes.binaries.separation_growth_rate.F90
Modules used: iso_varying_string
Used by: subroutine node_component_black_hole_standard_mass_accretion_rate subroutine node_component_black_hole_standard_output_properties subroutine node_component_black_hole_standard_rate_compute

function: black_hole_binary_separation_growth_rate
Description: Computes the separation growth rate of a black hole binary in units of Mpc/Gyr.
Code lines: 60
Contained by: module black_hole_binary_separations
Modules used: black_hole_binary_separations_null black_hole_binary_separations_standard
galacticus_error galacticus_nodes input_parameters

file: black_holes.binaries.separation_growth_rate.null.F90
Description: Contains a module which implements a black hole binary separation growth rate which is always zero.
Code lines: 54

module: black_hole_binary_separations_null
Description: Implements a black hole binary initial separation growth rate which is always zero.
Code lines: 32
Contained by: file black_holes.binaries.separation_growth_rate.null.F90
Used by: function black_hole_binary_separation_growth_rate
function: black_hole_binary_separation_growth_rate_null
Description: Returns a separation growth rate for a binary black hole that is always zero.
Code lines: 8
Contained by: module black_hole_binary_separations_null
Modules used: galacticus_nodes

subroutine: black_hole_binary_separation_growth_rate_null_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 9
Contained by: module black_hole_binary_separations_null
Modules used: iso_varying_string

file: black_holes.binaries.separation_growth_rate.standard.F90
Description: Contains a module which implements a black hole binary separation growth rate which follows a modified version of Volonteri et al. [2003], including terms for dynamical friction, hardening due to scattering of stars and emission of gravitational waves.
Code lines: 199

module: black_hole_binary_separations_standard
Description: Implements a black hole binary initial separation growth rate which follows a modified version of Volonteri et al. [2003], including terms for dynamical friction, hardening due to scattering of stars and emission of gravitational waves.
Code lines: 175
Contained by: file black_holes.binaries.separation_growth_rate.standard.F90
Used by: function black_hole_binary_separation_growth_rate

function: black_hole_binary_separation_growth_rate_standard
Description: Returns an initial separation growth rate for a binary black holes that follows a modified version of Volonteri et al. [2003].
Code lines: 116
Contained by: module black_hole_binary_separations_standard
Modules used: dark_matter_halo_scales galactic_structure_densities
galactic_structure_options galactic_structure_rotation_curve_gradients
galactic_structure_rotation_curves galactic_structure_velocity_dispersions
galacticus_display galacticus_error
galacticus_nodes numerical_constants_astronomical
numerical_constants_physical

subroutine: black_hole_binary_separation_growth_rate_standard_init
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 36
Contained by: module black_hole_binary_separations_standard
Modules used: input_parameters iso_varying_string

19.1. Program units
file: black_holes.binary_mergers.F90
Description: Contains a module which implements calculations of black hole binary mergers.
Code lines: 98

module: black_hole_binary_mergers
Description: Implements calculations of black hole binary mergers.
Code lines: 78
Contained by: file black_holes.binary_mergers.F90
Modules used: iso_varying_string
Used by: subroutine node_component_black_hole_simple_satellite_merging subroutine node_component_black_hole_standard_merge_black_holes subroutine node_component_black_hole_standard_satellite_merging

subroutine: black_hole_binary_merger
Description: Computes the effects of a black hole binary merger.
Code lines: 58
Contained by: module black_hole_binary_mergers
Modules used: black_hole_binary_mergers_rezzolla galacticus_error input_parameters

file: black_holes.binary_mergers.Rezzolla2008.F90
Description: Contains a module which implements calculations of black hole mass and spin resulting from binary mergers utilizing the approximations of Rezzolla et al. [2008].
Code lines: 92

module: black_hole_binary_mergers_rezzolla
Description: Implements calculations of black hole mass and spin resulting from binary mergers utilizing the approximations of Rezzolla et al. [2008].
Code lines: 71
Contained by: file black_holes.binary_mergers.Rezzolla2008.F90
Used by: subroutine black_hole_binary_merger_rezzolla

subroutine: black_hole_binary_merger_rezzolla
Description: Computes the mass and spin of a black hole resulting from a binary merger utilizing the approximations of Rezzolla et al. [2008].
Code lines: 46
Contained by: module black_hole_binary_mergers_rezzolla

subroutine: black_hole_binary_merger_rezzolla_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 9
Contained by: module black_hole_binary_mergers_rezzolla
Modules used: iso_varying_string

file: black_holes.fundamentals.F90
Description: Contains a module which implements fundamental properties of black holes.
Code lines: 557
module: black_hole_fundamentals
Description: Implements fundamental properties of black holes.
Code lines: 537
Contained by: file black_holes.fundamentals.F90
Modules used: galacticus_nodes
Used by:
  subroutine accretion_disk_adaf_tabulate
  function adaf_disk_jet_power
  function adaf_field_enhancement
  function adaf_gamma_phi
  function adaf_temperature
  function adaf_v
  function accretion_disk__jet_power__eddington
  function accretion_disk__jet_power__shakura_sunyaev
  function accretion_disk__radiative__efficiency_shakura_sunyaev
  function accretion_disk__switched_adaf__fraction
  function node_component_black_hole__simple_potential
  function node_component_black_hole__simple_rotation_curve
  function node_component_black_hole__standard_rotation_curve
  function node_component_black_hole__standard_rotation_curve_gradient

  subroutine node_component_black_hole__standard_mass_accretion_rate
  subroutine node_component_black_hole__standard_rotation_curve
  subroutine node_component_black_hole__standard_rotation_curve_gradient

  subroutine accretion_disk_adaf_tabulate
  function adaf_bh_jet_power
  function adaf_enthalpy__angular__momentum_product
  function adaf_fluid__angular__velocity
  function adaf_height
  function adaf_v

function: a1
Description: Return the function $A_1(j)$ that appears in the Kerr metric with spin $\text{blackHoleSpin}$.  
Code lines: 7
Contained by: module black_hole_fundamentals

function: a2
Description: Return the function $A_2(j)$ that appears in the Kerr metric with spin $\text{blackHoleSpin}$.  
Code lines: 7
Contained by: module black_hole_fundamentals

function: black_hole_eddington_accretion_rate
Description: Return the Eddington accretion rate (in $M_\odot\text{Gyr}^{-1}$) for the black hole in $\text{thisBlackHole}$.  
Code lines: 9
Contained by: module black_hole_fundamentals
Modules used: numerical_constants_astronomical numerical_constants_physical

interface: black_hole_frame_dragging_frequency
Code lines: 3
Contained by: module black_hole_fundamentals

function: black_hole_frame_dragging_frequency_node
Description: Returns the frame-dragging angular velocity in the Kerr metric.
function: black_hole_frame_dragging_frequency_spin
Description: Returns the frame-dragging angular velocity in the Kerr metric.
Code lines: 7
Contained by: module black_hole_fundamentals

function: black_hole_gravitational_radius
Description: Computes the gravitational radius (in Mpc) for the thisBlackHole.
Code lines: 8
Contained by: module black_hole_fundamentals
Modules used: numerical_constants_physical

interface: black_hole_horizon_radius
Code lines: 3
Contained by: module black_hole_fundamentals

function: black_hole_horizon_radius_node
Description: Return the radius of the horizon for a Kerr metric with dimensionless angular momentum j. The radius is in units of the gravitational radius.
Code lines: 30
Contained by: module black_hole_fundamentals
Modules used: galacticus_error

function: black_hole_horizon_radius_spin
Description: Return the radius of the horizon for a Kerr metric with dimensionless angular momentum j. The radius is in units of the gravitational radius.
Code lines: 8
Contained by: module black_hole_fundamentals

interface: black_hole_isco_radius
Code lines: 3
Contained by: module black_hole_fundamentals

function: black_hole_isco_radius_node
Description: Returns the radius (in physical or gravitational units and for a prograde or retorgrade orbit) of the innermost stable circular orbit for the black hole in thisBlackHole.
Code lines: 32
Contained by: module black_hole_fundamentals

function: black_hole_isco_radius_spin
Description: Returns the radius (in gravitational units and for a prograde or retorgrade orbit) of the innermost stable circular orbit for a black hole with spin blackHoleSpin.
Code lines: 41
Contained by: module black_hole_fundamentals
Modules used: galacticus_error

function: black_hole_isco_specific_angular_momentum
19.1. Program units

*Description:* Returns the specific angular momentum (in physical or gravitational units and for a prograde or retorgrade orbit) of the innermost stable circular orbit for the black hole in `thisBlackHole`.

*Code lines:* 42

*Contained by:* module `black_hole_fundamentals`

*Modules used:* `numerical_constants_physical`

**interface: black_hole_isco_specific_energy**

*Code lines:* 3

*Contained by:* module `black_hole_fundamentals`

**function: black_hole_isco_specific_energy_node**

*Description:* Returns the specific energy (in physical or gravitational units and for a prograde or retorgrade orbit) of the innermost stable circular orbit for the black hole in `thisNode`.

*Code lines:* 26

*Contained by:* module `black_hole_fundamentals`

*Modules used:* `numerical_constants_physical`

**function: black_hole_isco_specific_energy_spin**

*Description:* Returns the specific energy (in physical or gravitational units and for a prograde or retorgrade orbit) of the innermost stable circular orbit for a black hole of given `blackHoleSpin`.

*Code lines:* 23

*Contained by:* module `black_hole_fundamentals`

**interface: black_hole_metric_a_factor**

*Code lines:* 3

*Contained by:* module `black_hole_fundamentals`

**function: black_hole_metric_a_factor_node**

*Description:* Returns the \( A \) factor appearing in the Kerr metric for `thisBlackHole`.

*Code lines:* 32

*Contained by:* module `black_hole_fundamentals`

*Modules used:* `galacticus_error`

**function: black_hole_metric_a_factor_spin**

*Description:* Returns the \( A \) factor appearing in the Kerr metric for spin `blackHoleSpin`.

*Code lines:* 7

*Contained by:* module `black_hole_fundamentals`

**interface: black_hole_metric_d_factor**

*Code lines:* 3

*Contained by:* module `black_hole_fundamentals`

**function: black_hole_metric_d_factor_node**

*Description:* Returns the \( D \) factor appearing in the Kerr metric for `thisBlackHole`.

*Code lines:* 32

*Contained by:* module `black_hole_fundamentals`

*Modules used:* `galacticus_error`
function: black_hole_metric_d_factor_spin
Description: Returns the $D$ factor appearing in the Kerr metric for spin `blackHoleSpin`.
Code lines: 7
Contained by: module `black_hole_fundamentals`

interface: black_hole_rotational_energy_spin_down
Code lines: 3
Contained by: module `black_hole_fundamentals`

function: black_hole_rotational_energy_spin_down_node
Description: Wrapper function for `Black_Hole_Rotational_Energy_Spin_Down_Node` which takes a tree node as input.
Code lines: 13
Contained by: module `black_hole_fundamentals`

function: black_hole_rotational_energy_spin_down_spin
Description: Computes the spin down rate of a black hole due to extraction of rotational energy. Specifically, it returns the factor $S$ in the relation:

$$s = -S \frac{P_{\text{rotation}}}{M_{\bullet} c^2},$$

(19.1)

where $P_{\text{rotation}}$ is the power of rotational energy extraction and

$$S = \frac{\left(1 + \sqrt{1 - j^2} \right)^2 + j^2}{j},$$

(19.2)

for black hole spin $j$.
Code lines: 30
Contained by: module `black_hole_fundamentals`

interface: black_hole_static_radius
Code lines: 3
Contained by: module `black_hole_fundamentals`

function: black_hole_static_radius_node
Description: Return the radius of the static limit for a Kerr metric for the black hole in `thisBlackHole` and angle `theta`.
Code lines: 33
Contained by: module `black_hole_fundamentals`
Modules used: `galacticus_error`

function: black_hole_static_radius_spin
Description: Return the radius of the static limit for a Kerr metric for a black hole of given `blackHoleSpin` and angle `theta`.
Code lines: 18
Contained by: module `black_hole_fundamentals`
Modules used: `numerical_constants_math`
19.1. Program units

**file:** chemical.reaction_rates.F90
Description: Contains a module that implements calculations of chemical reaction rates.

**module:** chemical.reaction_rates
Description: Implements calculations of chemical reaction rates.

**subroutine:** chemical_reaction_rate
Description: Return chemical reaction rates at the given temperature for the specified set of chemical densities (in cm$^{-3}$) and radiation field. Units of the returned rates are cm$^{-3}$ s$^{-1}$.

**subroutine:** chemical_reaction_rates_initialize
Description: Initialize the chemical reaction rates module.

**file:** chemical.reaction_rates.hydrogen.F90
Description: Contains a module which implements calculations of chemical reaction rates for hydrogen using the fits from Abel et al. [1997] and Tegmark et al. [1997].
## 19. Source Code Documentation

<table>
<thead>
<tr>
<th>Description</th>
<th>Computes the rate (in units of cm$^{-3}$ s$^{-1}$) for the reaction $H_2^+ + \gamma \rightarrow 2H^+ + e^-$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code lines</td>
<td>49</td>
</tr>
<tr>
<td>Contained by</td>
<td>module <code>chemical_hydrogen_rates</code></td>
</tr>
<tr>
<td>Modules used</td>
<td><code>numerical_constants_physical</code> <code>radiation_structure</code> <code>numerical_constants_units</code></td>
</tr>
</tbody>
</table>

### subroutine: `chemical_hydrogen_rate_h2_gamma_to_h2plus_electron`

<table>
<thead>
<tr>
<th>Description</th>
<th>Computes the rate (in units of cm$^{-3}$ s$^{-1}$) for the reaction $H_2 + \gamma \rightarrow H_2^+ + e^-$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code lines</td>
<td>48</td>
</tr>
<tr>
<td>Contained by</td>
<td>module <code>chemical_hydrogen_rates</code></td>
</tr>
<tr>
<td>Modules used</td>
<td><code>numerical_constants_physical</code> <code>radiation_structure</code> <code>numerical_constants_units</code></td>
</tr>
</tbody>
</table>

### subroutine: `chemical_hydrogen_rate_h2_gamma_to_h2star_to_2h`

<table>
<thead>
<tr>
<th>Description</th>
<th>Computes the rate (in units of cm$^{-3}$ s$^{-1}$) for the reaction $H_2 + \gamma \rightarrow H_2^* \rightarrow 2H$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code lines</td>
<td>46</td>
</tr>
<tr>
<td>Contained by</td>
<td>module <code>chemical_hydrogen_rates</code></td>
</tr>
<tr>
<td>Modules used</td>
<td><code>numerical_constants_physical</code> <code>radiation_structure</code> <code>numerical_constants_units</code></td>
</tr>
</tbody>
</table>

### subroutine: `chemical_hydrogen_rate_h2_h_to_3h`

<table>
<thead>
<tr>
<th>Description</th>
<th>Computes the rate (in units of c$^{-3}$ s$^{-1}$) for the reaction $H_2 + H \rightarrow 3H$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code lines</td>
<td>54</td>
</tr>
<tr>
<td>Contained by</td>
<td>module <code>chemical_hydrogen_rates</code></td>
</tr>
<tr>
<td>Modules used</td>
<td><code>numerical_constants_physical</code> <code>radiation_structure</code> <code>numerical_constants_units</code></td>
</tr>
</tbody>
</table>

### subroutine: `chemical_hydrogen_rate_h2_hplus_to_h2plus_h`

<table>
<thead>
<tr>
<th>Description</th>
<th>Computes the rate (in units of c$^{-3}$ s$^{-1}$) for the reaction $H_2 + H^+ \rightarrow H_2^+ + H$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code lines</td>
<td>53</td>
</tr>
<tr>
<td>Contained by</td>
<td>module <code>chemical_hydrogen_rates</code></td>
</tr>
<tr>
<td>Modules used</td>
<td><code>numerical_constants_physical</code> <code>radiation_structure</code> <code>numerical_constants_units</code></td>
</tr>
</tbody>
</table>

### subroutine: `chemical_hydrogen_rate_h2plus_electron_to_2h`

<table>
<thead>
<tr>
<th>Description</th>
<th>Computes the rate (in units of c$^{-3}$ s$^{-1}$) for the reaction $H_2^+ + e^- \rightarrow 2H$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code lines</td>
<td>47</td>
</tr>
<tr>
<td>Contained by</td>
<td>module <code>chemical_hydrogen_rates</code></td>
</tr>
<tr>
<td>Modules used</td>
<td><code>radiation_structure</code></td>
</tr>
</tbody>
</table>

### subroutine: `chemical_hydrogen_rate_h2plus_gamma_to_2hplus_electron`

<table>
<thead>
<tr>
<th>Description</th>
<th>Computes the rate (in units of cm$^{-3}$ s$^{-1}$) for the reaction $H_2^+ + \gamma \rightarrow 2H^+ + e^-$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code lines</td>
<td>50</td>
</tr>
<tr>
<td>Contained by</td>
<td>module <code>chemical_hydrogen_rates</code></td>
</tr>
<tr>
<td>Modules used</td>
<td><code>numerical_constants_physical</code> <code>radiation_structure</code> <code>numerical_constants_units</code></td>
</tr>
</tbody>
</table>

### subroutine: `chemical_hydrogen_rate_h2plus_gamma_to_h_hplus`

<table>
<thead>
<tr>
<th>Description</th>
<th>Computes the rate (in units of c$^{-3}$ s$^{-1}$) for the reaction $H_2 + \gamma \rightarrow 2H^+ + e^-$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code lines</td>
<td>50</td>
</tr>
<tr>
<td>Contained by</td>
<td>module <code>chemical_hydrogen_rates</code></td>
</tr>
<tr>
<td>Modules used</td>
<td><code>numerical_constants_physical</code> <code>radiation_structure</code> <code>numerical_constants_units</code></td>
</tr>
</tbody>
</table>

### subroutine: `chemical_hydrogen_rate_h2plus_gamma_to_h_hplus`
19.1. Program units

**Description:** Computes the rate (in units of c$^{-3}$ s$^{-1}$) for the reaction H$_2^+$ + γ → H + H$^+$.  
**Code lines:** 54  
**Contained by:** module chemical_hydrogen_rates  
**Modules used:** numerical_constants_physical numerical_constants_units radiation_structure

**subroutine:** chemical_hydrogen_rate_h2plus_h_to_h2_hplus  
**Description:** Computes the rate (in units of c$^{-3}$ s$^{-1}$) for the reaction H$_2^+$ + H → H$_2$ + H$^+$.  
**Code lines:** 44  
**Contained by:** module chemical_hydrogen_rates  
**Modules used:** radiation_structure

**subroutine:** chemical_hydrogen_rate_h2plus_hminus_to_h2_h  
**Description:** Computes the rate (in units of c$^{-3}$ s$^{-1}$) for the reaction H$_2^+$ + H$^-$ → H$_2$ + H.  
**Code lines:** 46  
**Contained by:** module chemical_hydrogen_rates  
**Modules used:** radiation_structure

**subroutine:** chemical_hydrogen_rate_h_electron_to_hminus_photon  
**Description:** Computes the rate (in units of cm$^{-3}$ s$^{-1}$) for the reaction H + e$^-$ → H$^-$ + γ.  
**Code lines:** 44  
**Contained by:** module chemical_hydrogen_rates  
**Modules used:** radiation_structure

**subroutine:** chemical_hydrogen_rate_h_electron_to_hplus_2electron  
**Description:** Computes the rate (in units of c$^{-3}$ s$^{-1}$) for the reaction H + e$^-$ → H$^+$ + 2e$^-$.  
**Code lines:** 43  
**Contained by:** module chemical_hydrogen_rates  
**Modules used:** atomic_rates_ionization_collisonal radiation_structure

**subroutine:** chemical_hydrogen_rate_h_gamma_to_hplus_electron  
**Description:** Computes the rate (in units of cm$^{-3}$ s$^{-1}$) for the reaction H + γ → H$^+$ + e$^-$.  
**Code lines:** 49  
**Contained by:** module chemical_hydrogen_rates  
**Modules used:** numerical_constants_physical numerical_constants_units radiation_structure

**subroutine:** chemical_hydrogen_rate_h_hminus_to_h2_electron  
**Description:** Computes the rate (in units of cm$^{-3}$ s$^{-1}$) for the reaction H + H$^-$ → H$_2$ + e$^-$.  
**Code lines:** 43  
**Contained by:** module chemical_hydrogen_rates  
**Modules used:** radiation_structure

**subroutine:** chemical_hydrogen_rate_h_hplus_to_h2plus_photon  
**Description:** Computes the rate (in units of c$^{-3}$ s$^{-1}$) for the reaction H + H$^+$ → H$_2^+$ + γ.  
**Code lines:** 53  
**Contained by:** module chemical_hydrogen_rates  
**Modules used:** numerical_constants_physical numerical_constants_units
**radiation_structure**

**subroutine:** chemical_hydrogen_rate_hminus_electron_to_h_2electron  
*Description:* Computes the rate (in units of cm$^{-3}$ s$^{-1}$) for the reaction H$_2^+$ + H → H$_2$ + H$^+$.  
*Code lines:* 44  
*Contained by:* module chemical_hydrogen_rates  
*Modules used:* radiation_structure

**subroutine:** chemical_hydrogen_rate_hminus_gamma_to_h_electron  
*Description:* Computes the rate (in units of cm$^{-3}$ s$^{-1}$) for the reaction H$^-$ + γ → H + e$^-$.  
*Code lines:* 52  
*Contained by:* module chemical_hydrogen_rates  
*Modules used:* numerical_constants_physical numerical_constants_units radiation_structure

**subroutine:** chemical_hydrogen_rate_hminus_h_to_2h_electron  
*Description:* Computes the rate (in units of cm$^{-3}$ s$^{-1}$) for the reaction H$^-$ + H → 2H + e$.  
*Code lines:* 54  
*Contained by:* module chemical_hydrogen_rates  
*Modules used:* numerical_constants_physical numerical_constants_units radiation_structure

**subroutine:** chemical_hydrogen_rate_hminus_hplus_to_2h  
*Description:* Computes the rate (in units of cm$^{-3}$ s$^{-1}$) for the reaction H$_2^+$ + H → H$_2$ + H$^+$.  
*Code lines:* 44  
*Contained by:* module chemical_hydrogen_rates  
*Modules used:* radiation_structure

**subroutine:** chemical_hydrogen_rate_hminus_hplus_to_h2plus_electron  
*Description:* Computes the rate (in units of cm$^{-3}$ s$^{-1}$) for the reaction H$^-$ + H$^+$ → H$_2^+$ + e$.  
*Code lines:* 55  
*Contained by:* module chemical_hydrogen_rates  
*Modules used:* numerical_constants_physical numerical_constants_units radiation_structure

**subroutine:** chemical_hydrogen_rate_hplus_electron_to_h_photon  
*Description:* Computes the rate (in units of cm$^{-3}$ s$^{-1}$) for the reaction H$^+$ + e$^- → H + γ$.  
*Code lines:* 41  
*Contained by:* module chemical_hydrogen_rates  
*Modules used:* atomic_rates_recombination_radiative radiation_structure

**subroutine:** chemical_hydrogen_rates_compute  
*Description:* Compute rates of change of chemical abundances due to reactions involving chemical hydrogen species.  
*Code lines:* 58  
*Contained by:* module chemical_hydrogen_rates  
*Modules used:* galacticus_error radiation_structure

**subroutine:** chemical_hydrogen_rates_initialize
19.1. Program units

**Description:** Initializes the chemical hydrogen reaction network module.

**Code lines:** 56

**Contained by:** module `chemical_hydrogen_rates`

**Modules used:**
- `galacticus_error`
- `input_parameters`
- `iso_varying_string`

**function:** `cross_section_h2_gamma_to_2h`

**Description:** Compute the cross-section (in units of cm\(^2\)) for the reaction \(H_2 + \gamma \rightarrow 2H\) as given by Abel et al. [1997].

**Code lines:** 43

**Contained by:** module `chemical_hydrogen_rates`

**Modules used:**
- `numerical_constants_physical`
- `numerical_constants_units`

---

**function:** `cross_section_h2plus_gamma_to_h2plus_electron`

**Description:**.Compute the cross-section (in units of cm\(^2\)) for the reaction \(H_2^+ + \gamma \rightarrow H_2^+ + e^-\) as given by Abel et al. [1997].

---

**function:** `cross_section_h2_plus_gamma_to_2hplus_electron`

**Description:** Compute the cross-section (in units of cm\(^2\)) for the reaction \(H_2^+ + \gamma \rightarrow 2H^+ + e^-\) as given by Shapiro and Kang [1987].

**Code lines:** 19

**Contained by:** module `chemical_hydrogen_rates`

**Modules used:**
- `numerical_constants_physical`
- `numerical_constants_units`

---

**function:** `cross_section_h2_plus_gamma_to_hplus`

**Description:** Compute the cross-section (in units of cm\(^2\)) for the reaction \(H_2^+ + \gamma \rightarrow H + H^+\) as given by Shapiro and Kang [1987].

**Code lines:** 21

**Contained by:** module `chemical_hydrogen_rates`

**Modules used:**
- `numerical_constants_physical`
- `numerical_constants_units`

---

**function:** `cross_section_h_gamma_to_hplus_electron`

**Description:** Compute the cross-section (in units of cm\(^2\)) for the reaction \(H_2 + \gamma \rightarrow 2H^+\) as given by Abel et al. [1997].

**Code lines:** 11

**Contained by:** module `chemical_hydrogen_rates`

**Modules used:**
- `atomic_cross_sections_ionization_photo`

---

**function:** `cross_section_hminus_gamma_to_h_electron`

---
Description: Compute the cross-section (in units of cm$^2$) for the reaction $\text{H}^- + \gamma \rightarrow \text{H} + e^-$ using the fitting function given by Shapiro and Kang [1987], renormalized\(^a\) to match the results of Nascimento and Goddard [1977].

\(^a\)It seems unclear what units were used in Shapiro and Kang [1987], hence the recalibration.

Code lines: 22

Contained by: module `chemical_hydrogen_rates`
19.1. Program units

Modules used: numerical_constants_physical numerical_constants_units

function: h_electron_to_hminus_photon_rate_coefficient
Description: Computes the rate coefficient (in units of cm$^3$ s$^{-1}$) for the reaction H + e$^-$ → H$^-$ + γ.
Code lines: 29
Contained by: module chemical_hydrogen_rates

function: h_hminus_to_h2_electron_rate_coefficient
Description: Computes the rate coefficient (in units of c$^3$ s$^{-1}$) for the reaction H + H$^-$ → H$_2$ + e$^-$. 
Code lines: 21
Contained by: module chemical_hydrogen_rates

function: hminus_electron_to_h_2electron_rate_coefficient
Description: Computes the rate coefficient (in units of cm$^3$ s$^{-1}$) for the reaction H$^+$ + H → H$_2$ + H$^T$.
Code lines: 15
Contained by: module chemical_hydrogen_rates

function: hminus_hplus_to_2h_rate_coefficient
Description: Computes the rate coefficient (in units of c$^3$ s$^{-1}$) for the reaction H$^+$ + H → H$_2$ + H$^+$. 
Code lines: 7
Contained by: module chemical_hydrogen_rates

file: chemical.reaction_rates.null.F90
Description: Contains a module which implements a null calculation of chemical reaction rates.
Code lines: 56

module: chemical_reaction_rates_null
Description: Implements a null calculation of chemical reaction rates.
Code lines: 36
Contained by: file chemical.reaction_rates.null.F90
Used by: subroutine chemical_reaction_rate subroutine chemical_reaction_rates_null_initialize

subroutine: chemical_reaction_rates_null_compute
Description: Computes rates of change of chemical abundances due to reactions involving chemical hydrogen species.
Code lines: 11
Contained by: module chemical_reaction_rates_null
Modules used: chemical_abundances_structure radiation_structure

subroutine: chemical_reaction_rates_null_initialize
Description: Initializes the null chemical reaction network module.
Code lines: 7
Contained by: module chemical_reaction_rates_null
Modules used: iso_varying_string

file: chemical.reaction_rates.utilities.F90
Description: Contains a module that implements various useful utility functions for calculations of chemical abundances and rates.

Code lines: 39

**module: chemical_reaction_rates_utilities**

Description: Implements various useful utility functions for calculations of chemical abundances and rates.

Code lines: 19

Contained by: file chemical.reaction_rates.utilities.F90

Used by:
- function coldmodechemicalmasses
- function simplechemicalmasses
- function coldmodecoldmodefraction
- function simplechemicalmasses
- function cooling_radius_isothermal
- subroutine cooling_radius_solver_.initialize
- subroutine node_component_hot_halo_.standard formation
- subroutine node_component_hot_halo_.standard_outflow_return
- subroutine node_component_hot_halo_.standard_rate_compute

**function: chemicals_mass_to_density_conversion**

Description: Returns the conversion factor from mass of chemicals in \(M_\odot/M_{\text{atomic}}\) to number density in cm\(^{-3}\) assuming that the mass is distributed uniformly in a sphere of the given radius (in Mpc).

Code lines: 9

Contained by: module chemical_reaction_rates_utilities

Modules used: numerical_constants_astronomical

**file: chemical.state.CIE_file.F90**

Description: Contains a module which reads and interpolates a collisional ionization equilibrium chemical state from a file.

Code lines: 685

**module: chemical_states_cie_file**

Description: Reads and interpolates a collisional ionization equilibrium chemical state from a file.

Code lines: 665

Contained by: file chemical.state.CIE_file.F90

Modules used: fgsl

Used by:
- subroutine chemical_state_initialize
- subroutine chemical_densities_atomic_.cie_cloudy
- subroutine chemical_state_atomic_cie_.cloudy_create
- function electron_density_atomic_cie_.cloudy
- function electron_density_temperature_.log_slope_atomic_cie_cloudy

**subroutine: chemical_densities_cie_file**

Description: Return the densities of chemical species at the given temperature and hydrogen density for the specified set of abundances and radiation field. Units of the returned electron density are cm\(^{-3}\).

Code lines: 19

Contained by: module chemical_states_cie_file

Modules used: abundances_structure chemical_abundances_structure radiation_structure
subroutine: chemical_densities_cie_file_interpolate
Description: Compute the chemical state by interpolation in the tabulated data.
Code lines: 88
Contained by: module chemical_states_cie_file
Modules used: abundances_structure chemical_abundances_structure
                  io_xml iso_c_binding
                  numerical_constants_astronomical radiation_structure

subroutine: chemical_state_cie_chemicals_initialize
Description: Ensure that chemical indices have been found.
Code lines: 29
Contained by: module chemical_states_cie_file
Modules used: chemical_abundances_structure

function: chemical_state_cie_file_format_version
Description: Return the current file format version of CIE chemical state files.
Code lines: 6
Contained by: module chemical_states_cie_file

subroutine: chemical_state_cie_file_initialize
Description: Initializes the “CIE ionization state from file” module.
Code lines: 29
Contained by: module chemical_states_cie_file
Modules used: input_parameters

subroutine: chemical_state_cie_file_read
Description: Read in data from an chemical state file.
Code lines: 155
Contained by: module chemical_states_cie_file
Modules used: fox_dom galacticus_display
                  galacticus_error io_xml
                  memory_management numerical_comparison

subroutine: chemical_state_cie_file_read_initialize
Description: Ensure that the cooling data file has been read in.
Code lines: 20
Contained by: module chemical_states_cie_file

function: do_interpolation
Description: Perform the interpolation.
Code lines: 15
Contained by: module chemical_states_cie_file
Modules used: iso_c_binding

function: electron_density_cie_file
Description: Return the electron density by interpolating in tabulated CIE data read from a file.
Code lines: 16
Contained by: module chemical_states_cie_file
19. Source Code Documentation

 Modules used: abundances_structure radiation_structure

 function: electron_density_cie_file_interpolate
 Description: Compute the chemical state by interpolation in the tabulated data.
 Code lines: 77
 Contained by: module chemical_states_cie_file
 Modules used: abundances_structure io_xml
 iso_c_binding numerical_constants_astronomical
 radiation_structure

 function: electron_density_cie_file_logtemperature_interpolate
 Description: Compute the logarithmic gradient of the electron density with respect to temperature by interpolation in the tabulated data.
 Code lines: 80
 Contained by: module chemical_states_cie_file
 Modules used: abundances_structure io_xml
 iso_c_binding numerical_constants_astronomical
 radiation_structure

 function: electron_density_density_log_slope_cie_file
 Description: Return the logarithmic slope of the electron density with respect to density assuming atomic CIE as computed by CLOUDY.
 Code lines: 13
 Contained by: module chemical_states_cie_file
 Modules used: abundances_structure radiation_structure

 function: electron_density_temperature_log_slope_cie_file
 Description: Return the logarithmic slope of the electron density with respect to temperature by interpolating in tabulated CIE data read from a file.
 Code lines: 17
 Contained by: module chemical_states_cie_file
 Modules used: abundances_structure radiation_structure

 subroutine: get_interpolation
 Description: Determine the interpolating parameters.
 Code lines: 30
 Contained by: module chemical_states_cie_file
 Modules used: iso_c_binding numerical_interpolation

 file: chemical.state.F90
 Description: Contains a module that implements calculations of the chemical state.
 Code lines: 187

 module: chemical_states
 Description: Implements calculations of the chemical state.
 Code lines: 167
 Contained by: file chemical.state.F90
 Modules used: abundances_structure chemical_abundances_structure
 iso_varying_string radiation_structure
19.1. Program units

Used by:

- function coldmodechemicalmasses
- function coldmodecoldmodefraction
- function simplechemicalmasses
- function coldmodechemicalmasses
- subroutine cooling_function_cmb_compton
- subroutine cooling_function_density_slope_cmb_compton
- subroutine cooling_function_temperature_slope_cmb_compton
- subroutine node_component_hot_halo_standard_formation
- subroutine node_component_hot_halo_standard_outflow_return
- subroutine node_component_hot_halo_standard_rate_compute

subroutine: chemical_densities
Description: Return the densities of chemical species at the given temperature and hydrogen density for the specified set of abundances and radiation field. Units of the returned electron density are cm$^{-3}$.
Code lines: 16
Contained by: module chemical_states

subroutine: chemical_state_initialize
Description: Initialize the chemical state module.
Code lines: 55
Contained by: module chemical_states
Modules used: chemical_states_atomic_cie_cloudy chemical_states_cie_file galacticus_error input_parameters

function: electron_density
Description: Return the electron density at the given temperature and hydrogen density for the specified set of abundances and radiation field. Units of the returned electron density are cm$^{-3}$.
Code lines: 15
Contained by: module chemical_states

function: electron_density_density_log_slope
Description: Return the logarithmic gradient of electron density with respect to density at the given temperature and hydrogen density for the specified set of abundances and radiation field. Units of the returned electron density are cm$^{-3}$.
Code lines: 15
Contained by: module chemical_states

function: electron_density_temperature_log_slope
Description: Return the logarithmic gradient of electron density with temperature at the given temperature and hydrogen density for the specified set of abundances and radiation field. Units of the returned electron density are cm$^{-3}$.
Code lines: 15
Contained by: module chemical_states

file: chemical.state.atomic_CIE_cloudy.F90
Description: Contains a module which generates a tabulated atomic collisional ionization equilibrium ionization state using CLOUDY.
Code lines: 200
module: chemical_states_atomic_cie_cloudy
  Description: Generates a tabulated atomic collisional ionization equilibrium ionization state using CLOUDY.
  Code lines: 180
  Contained by: file chemical_state.atomic_CIE_cloudy.F90
  Modules used: iso_varying_string
  Used by: subroutine chemical_state_initialize

subroutine: chemical_densities_atomic_cie_cloudy
  Description: Return the densities of chemical species at the given temperature and hydrogen density for the specified set of abundances and radiation field. Units of the returned electron density are cm⁻³.
  Code lines: 20
  Contained by: module chemical_states_atomic_cie_cloudy
  Modules used: abundances_structure chemical_abundances_structure chemical_states_cie_file radiation_structure

subroutine: chemical_state_atomic_cie_cloudy_create
  Description: Create the chemical state.
  Code lines: 54
  Contained by: module chemical_states_atomic_cie_cloudy
  Modules used: abundances_structure chemical_states_cie_file galacticus_input_paths string_handling system_command

subroutine: chemical_state_atomic_cie_cloudy_initialize
  Description: Initializes the “atomic CIE ionization state from CLOUDY” module.
  Code lines: 18
  Contained by: module chemical_states_atomic_cie_cloudy

function: electron_density_atomic_cie_cloudy
  Description: Return the electron density assuming atomic CIE as computed by CLOUDY.
  Code lines: 17
  Contained by: module chemical_states_atomic_cie_cloudy
  Modules used: abundances_structure chemical_states_cie_file radiation_structure

function: electron_density_density_log_slope_atomic_cie_cloudy
  Description: Return the logarithmic slope of the electron density with respect to density assuming atomic CIE as computed by CLOUDY.
  Code lines: 13
  Contained by: module chemical_states_atomic_cie_cloudy
  Modules used: abundances_structure radiation_structure

function: electron_density_temperature_log_slope_atomic_cie_cloudy
  Description: Return the logarithmic slope of the electron density with respect to temperature assuming atomic CIE as computed by CLOUDY.
  Code lines: 17
  Contained by: module chemical_states_atomic_cie_cloudy
19.1. Program units

Modules used: abundances_structure chemical_states_cie_file
radiation_structure

file: constant.F90
Code lines: 262

module: constants_nswc
Code lines: 258
Contained by: file constant.F90
Used by: module incomplete_gamma

function: depsln
Code lines: 13
Contained by: module constants_nswc

function: dpmpar
Code lines: 33
Contained by: module constants_nswc

function: dxparg
Code lines: 23
Contained by: module constants_nswc

function: epsln
Code lines: 13
Contained by: module constants_nswc

function: exparg
Code lines: 23
Contained by: module constants_nswc

function: ipmpar
Code lines: 82
Contained by: module constants_nswc

function: spmpar
Code lines: 33
Contained by: module constants_nswc

file: cooling.cold_mode.infall_rate.F90
Description: Contains a module that implements calculations of the infall rate from the cold mode.
Code lines: 213

module: cooling_cold_mode_infall_rates
Description: Implements calculations of the infall rate from the cold mode.
Code lines: 193
Contained by: file cooling.cold_mode.infall_rate.F90
Modules used: galacticus_nodes iso_varying_string
Used by: subroutine count_properties subroutine establish_property_names
subroutine galacticus_merger_tree_output

subroutine node_component_hot_halo_cold_mode_rate_compute

subroutine: cooling_cold_mode_infall_output
Description: Store cold mode infall rate properties in the GALACTICUS output file buffers.
Code lines: 18
Contained by: module cooling_cold_mode_infall_rates
Modules used: kind_numbers

subroutine: cooling_cold_mode_infall_output_count
Description: Account for the number of cold mode infall rate properties to be written to the GALACTICUS output file.
Code lines: 13
Contained by: module cooling_cold_mode_infall_rates

subroutine: cooling_cold_mode_infall_output_initialize
Description: Initialize output in the cold mode infall rate module.
Code lines: 26
Contained by: module cooling_cold_mode_infall_rates
Modules used: input_parameters

subroutine: cooling_cold_mode_infall_output_names
Description: Set names of cold mode infall rate properties to be written to the GALACTICUS output file.
Code lines: 29
Contained by: module cooling_cold_mode_infall_rates
Modules used: numerical_constants_astronomical

function: cooling_cold_mode_infall_rate
Description: Return the cold mode infall rate for thisNode (in units of $M_{\odot}$ Gyr$^{-1}$).
Code lines: 11
Contained by: module cooling_cold_mode_infall_rates

subroutine: cooling_cold_mode_infall_rate_initialize
Description: Initialize the cooling rate module.
Code lines: 52
Contained by: module cooling_cold_mode_infall_rates
Modules used: cooling_cold_mode_infall_rates_dynamical_time, galacticus_error, dynamical_time, input_parameters

file: cooling.cold_mode.infall_rate.dynamical_time.F90
Description: Contains a module which implements a calculation of cold mode infall rates assuming infall on a dynamical timescale.
Code lines: 80

module: cooling_cold_mode_infall_rates_dynamical_time
Description: Implements a calculation of cold mode infall rates assuming infall on a dynamical timescale.
Code lines: 60
Contained by: file cooling.cold_mode.infall_rate.dynamical_time.F90
19.1. Program units

**Modules used:** galacticus_nodes

**Used by:** subroutine cooling_cold_mode_infall_rate_initialize

**function:** cooling_cold_mode_infall_rate_dynamical_time

**Description:** Computes the mass cooling rate in a hot gas halo assuming a fixed timescale for cooling.

**Code lines:** 12

**Contained by:** module cooling_cold_mode_infall_rates_dynamical_time

**Modules used:**
- dark_matter_halo_scales

**subroutine:** cooling_cold_mode_infall_rate_dynamical_time_initialize

**Description:** Initializes the “dynamical time” cold mode infall rate module.

**Code lines:** 29

**Contained by:** module cooling_cold_mode_infall_rates_dynamical_time

**Modules used:**
- galacticus_error
- input_parameters
- iso_varying_string

**file:** cooling.cooling_function.CIE_file.F90

**Description:** Contains a module which reads and interpolates a collisional ionization equilibrium cooling function from a file.

**Code lines:** 575

**module:** cooling_functions_cie_file

**Description:** Reads and interpolates a collisional ionization equilibrium cooling function from a file.

**Code lines:** 555

**Contained by:** file cooling.cooling_function.CIE_file.F90

**Modules used:**
- fgsl

**Used by:**
- function cooling_function
- function cooling_function_density_log_slope
- subroutine cooling_function_initialize
- function cooling_function_temperature_log_slope
- subroutine cooling_function_atomic_cie_cloudy
- subroutine cooling_function_temperature_slope_atomic_cie_cloudy

**subroutine:** cooling_function_cie_file

**Description:** Return the cooling function by interpolating in tabulated CIE data read from a file.

**Code lines:** 29

**Contained by:** module cooling_functions_cie_file

**Modules used:**
- abundances_structure
- chemical_abundances_structure
- radiation_structure

**function:** cooling_function_cie_file_format_version

**Description:** Return the current file format version of CIE cooling files.

**Code lines:** 6

**Contained by:** module cooling_functions_cie_file

**subroutine:** cooling_function_cie_file_initialize
19. Source Code Documentation

**Description:** Initializes the “CIE cooling function from file” module.

**Code lines:** 23

**Contained by:** module `cooling_functions_cie_file`

**Modules used:** `input_parameters`

**function:** `cooling_function_cie_file_interpolate`

**Description:** Compute the cooling function by interpolation in the tabulated data.

**Code lines:** 76

**Contained by:** module `cooling_functions_cie_file`

**Modules used:**
- `abundances_structure`
- `io_xml`
- `iso_c_binding`
- `numerical_constants_astronomical`
- `radiation_structure`

**function:** `cooling_function_cie_file_logtemperature_interpolate`

**Description:** Compute the logarithmic gradient of the cooling function with respect to temperature by interpolation in the tabulated data.

**Code lines:** 80

**Contained by:** module `cooling_functions_cie_file`

**Modules used:**
- `abundances_structure`
- `io_xml`
- `iso_c_binding`
- `numerical_constants_astronomical`
- `radiation_structure`

**subroutine:** `cooling_function_cie_file_read`

**Description:** Read in data from a cooling function file.

**Code lines:** 142

**Contained by:** module `cooling_functions_cie_file`

**Modules used:**
- `fox_dom`
- `galacticus_display`
- `galacticus_error`
- `io_xml`
- `memory_management`
- `numerical_comparison`

**subroutine:** `cooling_function_cie_file_read_initialize`

**Description:** Ensure that the cooling data file has been read in.

**Code lines:** 17

**Contained by:** module `cooling_functions_cie_file`

**subroutine:** `cooling_function_density_slope_cie_file`

**Description:** Return the logarithmic slope of the cooling function with respect to density.

**Code lines:** 30

**Contained by:** module `cooling_functions_cie_file`

**Modules used:**
- `abundances_structure`
- `chemical_abundances_structure`
- `radiation_structure`

**subroutine:** `cooling_function_temperature_slope_cie_file`

**Description:** Return the slope of the cooling function with respect to temperature by interpolating in tabulated CIE data read from a file.

**Code lines:** 34

**Contained by:** module `cooling_functions_cie_file`

**Modules used:**
- `abundances_structure`
- `chemical_abundances_structure`
- `radiation_structure`
function: do_interpolation
Description: Perform the interpolation.
Code lines: 14
Contained by: module cooling_functions_cie_file
Modules used: iso_c_binding

subroutine: get_interpolation
Description: Determine the interpolating parameters.
Code lines: 30
Contained by: module cooling_functions_cie_file
Modules used: iso_c_binding numerical_interpolation

file: cooling.cooling_function.CMB_Compton.F90
Description: Contains a module which computes the contribution to the cooling function due to Compton cooling off of the cosmic microwave background.
Code lines: 169

module: cooling_functions_cmb_compton
Description: Computes the contribution to the cooling function due to Compton cooling off of the cosmic microwave background.
Code lines: 148
Contained by: file cooling.cooling_function.CMB_Compton.F90
Modules used: iso_varying_string
Used by: function cooling_function function cooling_function_density_log_slope
subroutine cooling_function_initialize function cooling_function_temperature_log_slope

subroutine: cooling_function_cmb_compton
Description: Return the cooling function assuming atomic CIE as computed by CLOUDY.
Code lines: 34
Contained by: module cooling_functions_cmb_compton
Modules used: abundances_structure chemical_abundances_structure
chemical_states numerical_constants_physical
numerical_constants_units radiation_structure

subroutine: cooling_function_cmb_compton_initialize
Description: Initializes the “atomic CIE cooling function from CLOUDY” module.
Code lines: 13
Contained by: module cooling_functions_cmb_compton

subroutine: cooling_function_density_slope_cmb_compton
Description: Return the gradient with respect to density of cooling function assuming atomic CIE as computed by CLOUDY.
Code lines: 34
Contained by: module cooling_functions_cmb_compton
Modules used: abundances_structure chemical_abundances_structure
chemical_states radiation_structure
subroutine: cooling_function_temperature_slope_cmb_compton
Description: Return the cooling function assuming atomic CIE as computed by CLOUDY.
Code lines: 34
Contained by: module cooling_functions_cmb_compton
Modules used: abundances_structure chemical_abundances_structure
chemical_states radiation_structure

file: cooling.cooling_function.F90
Description: Contains a module that implements calculations of the cooling function.
Code lines: 340

module: cooling_functions
Description: Implements calculations of the cooling function.
Code lines: 320
Contained by: file cooling.cooling_function.F90
Modules used: abundances_structure chemical_abundances_structure
iso_varying_string radiation_structure
Used by: function coldmodecoldmodefraction function coldmodecoldmodefraction
function cooling_time_density_log_slope_simple function cooling_time_simple
function cooling_time_temperature_log_slope_simple

function: cooling_function
Description: Return the cooling function at the given temperature and hydrogen density for the specified set of abundances and radiation field. Units of the returned cooling function are the traditional ergs cm$^{-3}$ s$^{-1}$.
Code lines: 50
Contained by: module cooling_functions
Modules used: cooling_functions_atomic_cie_cloudy cooling_functions_cie_file
cooling_functions_cmb_compton cooling_functions_molecular_hydrogen_galli_palla

function: cooling_function_density_log_slope
Description: Return $d \ln \Lambda / d \ln \rho$ for a cooling function at the given temperature and hydrogen density for the specified set of abundances and radiation field.
Code lines: 60
Contained by: module cooling_functions
Modules used: cooling_functions_atomic_cie_cloudy cooling_functions_cie_file
cooling_functions_cmb_compton cooling_functions_molecular_hydrogen_galli_palla

subroutine: cooling_function_initialize
Description: Initialize the cooling function module.
Code lines: 68
Contained by: module cooling_functions
Modules used: cooling_functions_atomic_cie_cloudy cooling_functions_cie_file
19.1. Program units

subroutine:  cooling_function_not_matched
Code lines:  44
Contained by:  module  cooling_functions
Modules used:  galacticus_display

function:  cooling_function_temperature_log_slope
Description:  Return $d \ln \Lambda / d \ln T$ for a cooling function at the given temperature and hydrogen density for the specified set of abundances and radiation field.
Code lines:  60
Contained by:  module  cooling_functions
Modules used:  cooling_functions_atomic_cie_cloudy

file:  cooling.cooling_function.atomic_CIE_Cloudy.F90
Description:  Contains a module which generates a tabulated atomic collisional ionization equilibrium cooling function using CLOUDY.
Code lines:  233

module:  cooling_functions_atomic_cie_cloudy
Description:  Generates a tabulated atomic collisional ionization equilibrium cooling function using CLOUDY.
Code lines:  213
Contained by:  file  cooling.cooling_function.atomic_CIE_Cloudy.F90
Modules used:  iso_varying_string

subroutine:  cooling_function_atomic_cie_cloudy
Description:  Return the cooling function assuming atomic CIE as computed by CLOUDY.
Code lines:  30
Contained by:  module  cooling_functions_atomic_cie_cloudy
Modules used:  abundances_structure

subroutine:  cooling_function_atomic_cie_cloudy_create
Description:  Create the cooling function.
Code lines:  54
Contained by:  module  cooling_functions_atomic_cie_cloudy
Modules used:  abundances_structure
19. Source Code Documentation

subroutine: cooling_function_atomic_cie_cloudy_initialize
Description: Initializes the “atomic CIE cooling function from CLOUDY” module.
Code lines: 13
Contained by: module cooling_functions_atomic_cie_cloudy

subroutine: cooling_function_density_slope_atomic_cie_cloudy
Description: Return the gradient with respect to density of cooling function assuming atomic CIE as computed by CLOUDY.
Code lines: 30
Contained by: module cooling_functions_atomic_cie_cloudy
Modules used: abundances_structure chemical_abundances_structure radiation_structure

subroutine: cooling_function_temperature_slope_atomic_cie_cloudy
Description: Return the cooling function assuming atomic CIE as computed by CLOUDY.
Code lines: 34
Contained by: module cooling_functions_atomic_cie_cloudy
Modules used: abundances_structure chemical_abundances_structure cooling_functions_cie_file radiation_structure

file: cooling.cooling_function.molecular_hydrogen_Galli_Palla.F90
Description: Contains a module which computes the contribution to the cooling function from molecular hydrogen using the cooling function of Galli and Palla [1998].
Code lines: 381

module: cooling_functions_molecular_hydrogen_galli_palla
Description: Computes the contribution to the cooling function from molecular hydrogen using the cooling function of
Code lines: 360
Contained by: file cooling.cooling_function.molecular_hydrogen_Galli_Palla.F90
Modules used: iso_varying_string
Used by: function cooling_function function cooling_function_density_log_slope
function cooling_function_temperature_log_slope
subroutine cooling_function_initialize

subroutine: cooling_function_density_slope_molecular_hydrogen_gp
Description: Return the gradient with respect to density of the cooling function due to molecular hydrogen using the cooling function of Galli and Palla [1998].
Code lines: 42
Contained by: module cooling_functions_molecular_hydrogen_galli_palla
Modules used: abundances_structure chemical_abundances_structure radiation_structure

function: cooling_function_gp_h2plus_electron
Description: Compute the cooling function due to H^+_2−e^− interactions.
Code lines: 28
19.1. Program units

function: cooling_function_gp_h_h2
Description: Compute the cooling function due to H–H\textsubscript{2} interactions.
Code lines: 23

function: cooling_function_gp_h_h2plus
Description: Compute the cooling function due to H–H\textsubscript{2}\textsuperscript{+} interactions.
Code lines: 28

subroutine: cooling_function_molecular_hydrogen_gp
Description: Return the cooling function due to molecular hydrogen using the cooling function of Galli and Palla [1998] (which refers to the local thermodynamic equilibrium cooling function of Hollenbach and McKee [1979]). Cooling functions involving H\textsubscript{2}\textsuperscript{+} are computed using polynomial fits to the results of Suchkov and Shchekinov [1978] found by Andrew Benson by measuring curves from the original paper.
Code lines: 35

subroutine: cooling_function_molecular_hydrogen_gp_initialize
Description: Initializes the “molecular hydrogen Galli & Palla” cooling function module.
Code lines: 21

subroutine: cooling_function_temperature_slope_molecular_hydrogen_gp
Description: Return the gradient with respect to temperature of the cooling function due to molecular hydrogen using the cooling function of Galli and Palla [1998].
Code lines: 78

subroutine: number_density_critical_over_number_density_hydrogen
Description: Compute the ratio of critical number density to the hydrogen number density for use in molecular hydrogen cooling functions.
Code lines: 38

file: cooling.cooling_radius.F90
Description: Contains a module that implements calculations of the cooling radius.
module: cooling_radii
Description: Implements calculations of the cooling radius.
Code lines: 222
Contained by: file cooling.cooling_radius.F90
Modules used: 

galacticus_nodes
cooling_radii_isothermal
iso_varying_string

Used by:
function infall_radius_cooling_freefall
function infall_radius_cooling_radius

function infall_radius_cooling_freefall
description: Return the cooling radius for thisNode (in units of Mpc).
Code lines: 12
Contained by: module cooling_radii

function: cooling_radius_growth_rate
description: Return the rate at which the cooling radius grows for thisNode (in units of Mpc/Gyr).
Code lines: 12
Contained by: module cooling_radii

subroutine: cooling_radius_hot_halo_output
description: Store hot halo properties in the GALACTICUS output file buffers.
Code lines: 18
Contained by: module cooling_radii
Modules used: kind_numbers

subroutine: cooling_radius_hot_halo_output_count
description: Account for the number of hot halo cooling properties to be written to the GALACTICUS output file.
Code lines: 13
Contained by: module cooling_radii

subroutine: cooling_radius_hot_halo_output_names
description: Set names of hot halo properties to be written to the GALACTICUS output file.
Code lines: 29
Contained by: module cooling_radii
Modules used: numerical_constants_astronomical

subroutine: cooling_radius_initialize
description: Initialize the cooling radius module.
Code lines: 43
Contained by: module cooling_radii
19.1. Program units

**Modules used:** galacticus_error

**subroutine:** cooling_radius_output_initialize

*Description:* Initialize output in the cooling radius module.

*Code lines:* 27

*Contained by:* module cooling_radii

*Modules used:* input_parameters

**file:** cooling.cooling_radius.isothermal_profile.F90

*Description:* Contains a module which implements a calculation of cooling radius appropriate for an isothermal halo, assuming collisional ionization equilibrium such that cooling time scales as inverse density.

*Code lines:* 209

**module:** cooling_radii_isothermal

*Description:* Implements calculation of cooling radius appropriate for an isothermal halo, assuming collisional ionization equilibrium such that cooling time scales as inverse density.

*Code lines:* 187

*Contained by:* file cooling.cooling_radius.isothermal_profile.F90

*Modules used:* abundances_structure chemical_abundances_structure
galacticus_nodes kind_numbers
radiation_structure

*Used by:* module cooling_radii subroutine galacticus_calculations_reset

**function:** cooling_radius_growth_rate_isothermal

*Description:* Return the growth rate of the cooling radius in the “isothermal” model in Mpc/Gyr.

*Code lines:* 42

*Contained by:* module cooling_radii_isothermal

*Modules used:* cooling_times_available dark_matter_halo_scales

**function:** cooling_radius_isothermal

*Description:* Return the cooling radius in the isothermal model.

*Code lines:* 69

*Contained by:* module cooling_radii_isothermal

*Modules used:* chemical_reaction_rates_utilities cooling_times
cooling_times_available dark_matter_halo_scales
hot_halo_mass_distributions hot_halo_temperature_profile

**subroutine:** cooling_radius_isothermal_initialize

*Description:* Initializes the “isothermal” cooling radius module.

*Code lines:* 16

*Contained by:* module cooling_radii_isothermal

*Modules used:* iso_varying_string

**subroutine:** cooling_radius_isothermal_reset

*Description:* Reset the cooling radius calculation.

*Code lines:* 12

*Contained by:* module cooling_radii_isothermal
file: `cooling.cooling_radius.simple.F90`

*Description:* Contains a module which implements a simple cooling radius calculation (finds the radius at which the time available for cooling equals the cooling time).

*Code lines:* 292

**module: cooling_radii_simple**

*Description:* Implements a simple cooling radius calculation (finds the radius at which the time available for cooling equals the cooling time).

*Code lines:* 271

*Contained by:* file `cooling.cooling_radius.simple.F90`

*Modules used:*
- abundances_structure
- chemical_abundances_structure
- galacticus_nodes
- hot_halo_mass_distributions
- kind_numbers
- radiation_structure

*Used by:* module `cooling_radii` subroutine `galacticus_calculations_reset`

**function: cooling_radius_growth_rate_simple**

*Description:* Return the growth rate of the cooling radius in the “simple” model in Mpc/Gyr.

*Code lines:* 64

*Contained by:* module `cooling_radii_simple`

*Modules used:*
- cooling_times
- cooling_times_available
- hot_halo_temperature_profile

**function: cooling_radius_root**

*Description:* Root function which evaluates the difference between the cooling time at radius and the time available for cooling.

*Code lines:* 16

*Contained by:* module `cooling_radii_simple`

*Modules used:*
- cooling_times
- hot_halo_temperature_profile

**function: cooling_radius_simple**

*Description:* Return the cooling radius in the simple model.

*Code lines:* 61

*Contained by:* module `cooling_radii_simple`

*Modules used:*
- cooling_times_available
- root_finder

**subroutine: cooling_radius_simple_initialize**

*Description:* Initializes the “simple” cooling radius module.

*Code lines:* 21

*Contained by:* module `cooling_radii_simple`

*Modules used:*
- array_utilities
- galacticus_error
- iso_varying_string
- string_handling

**subroutine: cooling_radius_simple_reset**

*Description:* Reset the cooling radius calculation.

*Code lines:* 12

*Contained by:* module `cooling_radii_simple`
subroutine: cooling_radius_solver_initialize
Description: Initialize the abundances, chemical properties and radiation field for thisNode for use in cooling radius calculations.
Code lines: 36
Contained by: module cooling_radii_simple
Modules used: chemical_reaction_rates_utilities

Description: Contains a module which implements a Cole et al. [2000] cooling rate calculation.
Code lines: 89

module: cooling_rates_cole2000
Description: Implements a Cole et al. [2000] cooling rate calculation.
Code lines: 69
Used by: subroutine cooling_rate_initialize

function: cooling_rate_cole2000
Description: Computes the mass cooling rate in a hot gas halo utilizing the Cole et al. [2000] method. This is based on the properties of the halo at formation time, and gives a zero cooling rate when the cooling radius exceeds the virial radius.
Code lines: 44
Contained by: module cooling_rates_cole2000
Modules used: cooling_infall_radii galacticus_nodes hot_halo_mass_distributions numerical_constants_math

subroutine: cooling_rate_cole2000_initialize
Description: Initializes the “Cole et al. (2000)” cooling rate module.
Code lines: 10
Contained by: module cooling_rates_cole2000
Modules used: iso_varying_string

file: cooling.cooling_rate.F90
Description: Contains a module that implements calculations of the cooling rate.
Code lines: 256

module: cooling_rates
Description: Implements calculations of the cooling rate.
Code lines: 236
Contained by: file cooling.cooling_rate.F90
Modules used: galacticus_nodes iso_varying_string
Used by: subroutine count_properties subroutine establish_property_names subroutine node_component_hot_halo_output subroutine node_component_hot_halo_standard_rate subroutine node_component_mass_flow_statistics_standard_rate subroutine standard_cooling_rate subroutine very_simple_cooling_rate
function: cooling_rate
Description: Return the cooling rate for thisNode (in units of $M_\odot$ Gyr$^{-1}$).
Code lines: 37
Contained by: module cooling_rates
Modules used: cooling_rates_modifier_cut_off

subroutine: cooling_rate_hot_halo_output
Description: Store hot halo properties in the GALACTICUS output file buffers.
Code lines: 18
Contained by: module cooling_rates
Modules used: kind_numbers

subroutine: cooling_rate_hot_halo_output_count
Description: Account for the number of hot halo cooling properties to be written to the GALACTICUS output file.
Code lines: 13
Contained by: module cooling_rates

subroutine: cooling_rate_hot_halo_output_names
Description: Set names of hot halo properties to be written to the GALACTICUS output file.
Code lines: 29
Contained by: module cooling_rates
Modules used: numerical_constants_astronomical

subroutine: cooling_rate_initialize
Description: Initialize the cooling rate module.
Code lines: 60
Contained by: module cooling_rates
Modules used: cooling_rates_cole2000 cooling_rates_simple
cooling_rates_simple_scaling cooling_rates_white_frenk
cooling_rates_zero galacticus_error
input_parameters

subroutine: cooling_rate_output_initialize
Description: Initialize output in the cooling rate module.
Code lines: 26
Contained by: module cooling_rates
Modules used: input_parameters

file: cooling.cooling_rate.White-Frenk.F90
Description: Contains a module which implements a White and Frenk [1991] cooling rate calculation.
Code lines: 113

module: cooling_rates_white_frenk
Description: Implements a White and Frenk [1991] cooling rate calculation.
Code lines: 93
Contained by: file cooling.cooling_rate.White-Frenk.F90
19.1. Program units

**Modules used:** galacticus_nodes
**Used by:** subroutine cooling_rate_initialize

**function:** cooling_rate_white_frenk
**Description:** Computes the mass cooling rate in a hot gas halo utilizing the White and Frenk [1991] method.
**Code lines:** 44
**Contained by:** module cooling_rates_white_frenk
**Modules used:** cooling_infall_radii, dark_matter_halo_scales, hot_halo_mass_distributions, numerical_constants_math

**subroutine:** cooling_rate_white_frenk_initialize
**Description:** Initializes the “White-Frenk1991” cooling rate module.
**Code lines:** 30
**Contained by:** module cooling_rates_white_frenk
**Modules used:** array_utilities, galacticus_error, input_parameters, iso_varying_string

**file:** cooling.cooling_rate.modifier.cut_off.F90
**Description:** Contains a module which implements a cut off in the cooling rate at given redshift and virial velocity.
**Code lines:** 137

**module:** cooling_rates_modifier_cut_off
**Description:** Implements a cut off in the cooling rate at given redshift and virial velocity.
**Code lines:** 117
**Contained by:** file cooling.cooling_rate.modifier.cut_off.F90
**Used by:** function cooling_rate

**subroutine:** cooling_rate_modifier_cut_off
**Description:** Modify cooling rates by truncating them to zero below a given redshift and virial velocity.
**Code lines:** 94
**Contained by:** module cooling_rates_modifier_cut_off
**Modules used:** cosmology_functions, dark_matter_halo_scales, galacticus_error, galacticus_nodes, input_parameters, iso_varying_string

**file:** cooling.cooling_rate.simple.F90
**Description:** Contains a module which implements a simple cooling rate calculation in which the cooling rate equals the mass of hot gas divided by a fixed timescale.
**Code lines:** 78

**module:** cooling_rates_simple
**Description:** Implements a simple cooling rate calculation in which the cooling rate equals the mass of hot gas divided by a fixed timescale.
**Code lines:** 57
**Contained by:** file cooling.cooling_rate.simple.F90
**Modules used:** galacticus_nodes
**Used by:** subroutine cooling_rate_initialize
function: cooling_rate_simple
Description: Computes the mass cooling rate in a hot gas halo assuming a fixed timescale for cooling.
Code lines: 9
contained by: module cooling_rates_simple

subroutine: cooling_rate_simple_initialize
Description: Initializes the "simple" cooling rate module.
Code lines: 28
Contained by: module cooling_rates_simple
Modules used: galacticus_error input_parameters iso_varying_string

file: cooling.cooling_rate.simple_scaling.F90
Description: Contains a module which implements a simple cooling rate calculation in which the cooling rate equals the mass of hot gas divided by a timescale which is a function of halo mass and redshift.
Code lines: 141

module: cooling_rates_simple_scaling
Description: Implements a simple cooling rate calculation in which the cooling rate equals the mass of hot gas divided by a timescale which is a function of halo mass and redshift.
Code lines: 120
Contained by: file cooling.cooling_rate.simple_scaling.F90
Modules used: galacticus_nodes
Used by: subroutine cooling_rate_initialize

function: cooling_rate_simple_scaling
Description: Computes the mass cooling rate in a hot gas halo assuming a fixed timescale for cooling.
Code lines: 25
Contained by: module cooling_rates_simple_scaling
Modules used: cosmology_functions

subroutine: cooling_rate_simple_scaling_initialize
Description: Initializes the "simple scaling" cooling rate module.
Code lines: 75
Contained by: module cooling_rates_simple_scaling
Modules used: array_utilities galacticus_error input_parameters iso_varying_string

file: cooling.cooling_rate.zero.F90
Description: Contains a module which implements a zero cooling rate calculation.
Code lines: 52

module: cooling_rates_zero
Description: Implements zero cooling rate calculation.
Code lines: 32
Contained by: file cooling.cooling_rate.zero.F90
Used by: subroutine cooling_rate_initialize
function: cooling_rate_zero
  Description: Returns a zero mass cooling rate in a hot gas halos.
  Code lines: 8
  Contained by: module cooling_rates_zero
  Modules used: galacticus_nodes

subroutine: cooling_rate_zero_initialize
  Description: Initializes the “zero” cooling rate module.
  Code lines: 9
  Contained by: module cooling_rates_zero
  Modules used: iso_varying_string

file: cooling.cooling_time.F90
  Description: Contains a module that implements calculations of the cooling time.
  Code lines: 158

module: cooling_times
  Description: Implements calculations of the cooling function.
  Code lines: 138
  Contained by: file cooling.cooling_time.F90
  Modules used: abundances_structure chemical_abundances_structure cooling_times_simple iso_varying_string radiation_structure
  Used by: function cooling_radius_isothermal function cooling_radius_growth_rate_ simple function cooling_radius_root

function: cooling_time
  Description: Return the cooling time at the given temperature and density for the specified set of abundances and radiation field. Units of the returned cooling time are the Gyr.
  Code lines: 16
  Contained by: module cooling_times

function: cooling_time_density_log_slope
  Description: Return the logarithmic slope of the cooling time-density relation.
  Code lines: 15
  Contained by: module cooling_times

subroutine: cooling_time_initialize
  Description: Initialize the cooling time module.
  Code lines: 41
  Contained by: module cooling_times
  Modules used: galacticus_error input_parameters

function: cooling_time_temperature_log_slope
  Description: Return the logarithmic slope of the cooling time-temperature relation.
  Code lines: 15
  Contained by: module cooling_times
19. Source Code Documentation

**file: cooling.cooling_time.simple.F90**

*Description:* Contains a module which implements a simple cooling time calculation (based on the ratio of the thermal energy density to the volume cooling rate).

*Code lines:* 138

**module: cooling_times_simple**

*Description:* Implements a simple cooling time calculation (based on the ratio of the thermal energy density to the volume cooling rate).

*Code lines:* 117
*Contained by:* file cooling.cooling_time.simple.F90
*Used by:* module cooling_times

**function: cooling_time_density_log_slope_simple**

*Description:* Return $d \ln t_{cool}/d \ln \rho$ for gas at the given temperature (in Kelvin), density (in $M_\odot$ Mpc$^{-3}$), composition specified by gasAbundances and experiencing a radiation field as described by radiation.

*Code lines:* 15
*Contained by:* module cooling_times_simple
*Modules used:* abundances_structure chemical_abundances_structure cooling_functions radiation_structure

**function: cooling_time_simple**

*Description:* Compute the cooling time (in Gyr) for gas at the given temperature (in Kelvin), density (in $M_\odot$ Mpc$^{-3}$), composition specified by gasAbundances and experiencing a radiation field as described by radiation.

*Code lines:* 38
*Contained by:* module cooling_times_simple
*Modules used:* abundances_structure chemical_abundances_structure chemical_states cooling_functions numerical_constants_astronomical numerical_constants_physical radiation_structure

**subroutine: cooling_time_simple_initialize**

*Description:* Initializes the “simple” cooling time module.

*Code lines:* 27
*Contained by:* module cooling_times_simple
*Modules used:* input_parameters iso_varying_string

**function: cooling_time_temperature_log_slope_simple**

*Description:* Return $d \ln t_{cool}/d \ln T$ for gas at the given temperature (in Kelvin), density (in $M_\odot$ Mpc$^{-3}$), composition specified by gasAbundances and experiencing a radiation field as described by radiation.

*Code lines:* 15
*Contained by:* module cooling_times_simple
*Modules used:* abundances_structure chemical_abundances_structure cooling_functions radiation_structure

**file: cooling.freefall_radii.F90**
Description: Contains a module that implements calculations of the freefall radius in cooling calculations.

Code lines: 128

**module: freefall_radii**
Description: Implements calculations of the freefall radius in cooling calculations.
Code lines: 108
Contained by: file cooling.freefall_radii.F90
Modules used: freefall_radii_dark_matter_halo galacticus_nodes

Used by: function infall_radius_cooling_freefall

function: freefall_radius
Description: Return the freefall radius for cooling calculations for thisNode (in units of Mpc).
Code lines: 12
Contained by: module freefall_radii

function: freefall_radius_growth_rate
Description: Return the rate at which the freefall radius for cooling calculations grows for thisNode (in units of Mpc/Gyr).
Code lines: 12
Contained by: module freefall_radii

subroutine: freefall_radius_initialize
Description: Initialize the cooling radius module.
Code lines: 41
Contained by: module freefall_radii

Modules used: galacticus_error input_parameters

file: cooling.freefall_radii.dark_matter_halo.F90
Description: Contains a module which implements a simple cooling radius calculation (finds the radius at which the time available for cooling equals the cooling time).
Code lines: 93

module: freefall_radii_dark_matter_halo
Description: Implements a simple cooling radius calculation (finds the radius at which the time available for cooling equals the cooling time).
Code lines: 72
Contained by: file cooling.freefall_radii.dark_matter_halo.F90
Used by: module freefall_radii

function: freefall_radius_dark_matter_halo
Description: Return the freefall radius in the “dark matter halo” model.
Code lines: 19
Contained by: module freefall_radii_dark_matter_halo
Modules used: cooling_freefall_times_available dark_matter_profiles galacticus_nodes

subroutine: freefall_radius_dark_matter_halo_initialize
19. Source Code Documentation

**Description:** Initializes the “darkMatterHalo” freefall radius module.
**Code lines:** 13
**Contained by:** module `freefall_radii_dark_matter_halo`
**Modules used:** `iso_varying_string`

**function:** `freefall_radius_growth_rate_dark_matter_halo`
**Description:** Return the growth rate of the freefall radius in the “dark matter halo” model in Mpc/Gyr.
**Code lines:** 22
**Contained by:** module `freefall_radii_dark_matter_halo`
**Modules used:**
- `cooling_freefall_times_available`
- `dark_matter_profiles`
- `galacticus_nodes`

**file:** `cooling.freefall_time_available.F90`
**Description:** Contains a module that implements calculations of the time available for freefall in cooling calculations.
**Code lines:** 128
**module:** `cooling_freefall_times_available`
**Description:** Implements calculations of the time available for freefall in cooling calculations.
**Code lines:** 108
**Contained by:** file `cooling.freefall_time_available.F90`
**Modules used:**
- `freefall_times_available_halo` - `galacticus_nodes`
- `formation`
- `iso_varying_string`
**Used by:** function `freefall_radius_dark_matter_halo` - function `freefall_radius_growth_rate_dark_matter_halo`

**function:** `cooling_freefall_time_available`
**Description:** Return the time available for freefall in cooling calculations in thisNode.
**Code lines:** 12
**Contained by:** module `cooling_freefall_times_available`

**interface:** `cooling_freefall_time_available_get_template`
**Code lines:** 5
**Contained by:** module `cooling_freefall_times_available`

**function:** `cooling_freefall_time_available_get_template`
**Code lines:** 3
**Contained by:** interface `cooling_freefall_time_available_get_template`

**function:** `cooling_freefall_time_available_increase_rate`
**Description:** Return the rate at which the time available for freefall in cooling calculations increases in thisNode.
**Code lines:** 12
**Contained by:** module `cooling_freefall_times_available`

**subroutine:** `cooling_freefall_time_available_initialize`
**Description:** Initializes the freefall time available in cooling calculations module.
**Code lines:** 41
19.1. Program units

**Contained by:** module `cooling_freefall_times_available`  
**Modules used:** `galacticus_error`  
**Description:** Contains a module which implements the Cole et al. [2000] method for computing the time available for freefall in cooling calculations in hot halos.  
**Code lines:** 78

**file:** `cooling.freefall_time_available.halo_formation.F90`  
**Description:** Contains a module which implements the Cole et al. [2000] method for computing the time available for freefall in cooling calculations in hot halos.  
**Code lines:** 78

**module:** `freefall_times_available_halo_formation`  
**Description:** Implements the Cole et al. [2000] method for computing the time available for freefall in cooling calculations in hot halos.  
**Code lines:** 57  
**Contained by:** file `cooling.freefall_time_available.halo_formation.F90`  
**Modules used:** `galacticus_nodes`  
**Used by:** module `cooling_freefall_times_available`  
**Description:** Computes the time available for freefall using the Cole et al. [2000] method. Specifically, the time available is assumed to be the time since the halo formation event.  
**Code lines:** 12

**function:** `freefall_time_available_halo_formation`  
**Description:** Compute the time available for freefall using the Cole et al. [2000] method. Specifically, the time available is assumed to be the time since the halo formation event.  
**Code lines:** 12  
**Contained by:** module `freefall_times_available_halo_formation`  
**subroutine:** `freefall_time_available_halo_formation_initialize`  
**Description:** Initialize the Cole et al. [2000] freefall time available module.  
**Code lines:** 17  
**Contained by:** module `freefall_times_available_halo_formation`  
**Modules used:** `galacticus_error`  
**iso_varying_string`  
**function:** `freefall_time_available_increase_rate_halo_formation`  
**Description:** Compute the rate of increase of the time available for freefall using the Cole et al. [2000] method. We return a rate of 1.  
**Code lines:** 9  
**Contained by:** module `freefall_times_available_halo_formation`  

**file:** `cooling.infall_radius.F90`  
**Description:** Contains a module that implements calculations of the infall radius for cooling calculations.  
**Code lines:** 130

**module:** `cooling_infall_radii`  
**Description:** Implements calculations of the infall radius for cooling calculations.  
**Code lines:** 110  
**Contained by:** file `cooling.infall_radius.F90`  
**Modules used:** `galacticus_nodes`  
**infall_radii_cooling_freefall`  
**iso_varying_string`  
**Used by:** function `cooling_rate_cole2000`  
**function** `cooling_rate_white_frenk`  
**subroutine** `node_component_hot_halo_cold_mode_push_to_cooling_pipes`  
**subroutine** `node_component_hot_halo_standard_push_to_cooling_pipes`
function: infall_radius
Description: Return the infall radius for thisNode (in units of Mpc).
Code lines: 12
Contained by: module cooling_infall_radii

function: infall_radius_growth_rate
Description: Return the rate at which the infall radius grows for thisNode (in units of Mpc/Gyr).
Code lines: 12
Contained by: module cooling_infall_radii

subroutine: infall_radius_initialize
Description: Initialize the infall radius module.
Code lines: 42
Contained by: module cooling_infall_radii
Modules used: galacticus_error input_parameters

file: cooling.infall_radius.cooling_and_freefall.F90
Description: Contains a module which implements an infall radius calculation in which the infall radius is the smaller of the cooling and freefall radii.
Code lines: 111

module: infall_radii_cooling_freefall
Description: Implements an infall radius calculation in which the infall radius is the smaller of the cooling and freefall radii.
Code lines: 90
Contained by: file cooling.infall_radius.cooling_and_freefall.F90
Used by: module cooling_infall_radii

function: infall_radius_cooling_freefall
Description: Return the infall radius in the “cooling and freefall” model in Mpc/Gyr.
Code lines: 30
Contained by: module infall_radii_cooling_freefall
Modules used: cooling_radii dark_matter_halo_scales freefall_radii galacticus_nodes

subroutine: infall_radius_cooling_freefall_initialize
Description: Initializes the “cooling and freefall” infall radius module.
Code lines: 13
Contained by: module infall_radii_cooling_freefall
Modules used: iso_varying_string

function: infall_radius_growth_rate_cooling_freefall
Description: Return the growth rate of the infall radius in the “cooling and freefall” model in Mpc/Gyr.
Code lines: 30
Contained by: module infall_radii_cooling_freefall
Modules used: cooling_radii dark_matter_halo_scales freefall_radii galacticus_nodes
file: cooling.infall_radius.cooling_radius.F90
Description: Contains a module which implements a simple infall radius calculation, simply assuming that the infall radius equals the cooling radius.
Code lines: 68

module: infall_radii_cooling_radius
Description: Implements a simple infall radius calculation, simply assuming that the infall radius equals the cooling radius.
Code lines: 48
Contained by: file cooling.infall_radius.cooling_radius.F90
Used by: module cooling_infall_radii

function: infall_radius_cooling_radius
Description: Return the growth rate of the infall radius in the “cooling radius” model in Mpc/Gyr.
Code lines: 9
Contained by: module infall_radii_cooling_radius
Modules used: cooling_radii galacticus_nodes

subroutine: infall_radius_cooling_radius_initialize
Description: Initializes the “cooling radius” infall radius module.
Code lines: 13
Contained by: module infall_radii_cooling_radius
Modules used: iso_varying_string

function: infall_radius_growth_rate_cooling_radius
Description: Return the growth rate of the infall radius in the “cooling radius” model in Mpc/Gyr.
Code lines: 9
Contained by: module infall_radii_cooling_radius
Modules used: cooling_radii galacticus_nodes

file: cooling.specific_angular_momentum.F90
Description: Contains a module that implements calculations of the specific angular momentum of cooling gas.
Code lines: 110

module: cooling_specific_angular_momenta
Description: Implements calculations of the specific angular momentum of cooling gas.
Code lines: 90
Contained by: file cooling.specific_angular_momentum.F90
Modules used: cooling_specific_angular_momenta_- cooling_specific_angular_momenta_mean constant_rotation galacticus_nodes iso_varying_string
Used by: subroutine node_component_hot_halo_- subroutine node_component_hot_halo_- cold_mode_push_to_cooling_pipes standard_push_to_cooling_pipes

function: cooling_specific_angular_momentum
Description: Return the specific angular momentum (in units of km/s Mpc) of cooling gas in thisNode.
Code lines: 13
19. Source Code Documentation

Contained by: module `cooling_specific-angular-momenta`

**subroutine:** `cooling_specific-angular-momentum_initialize`
*Description:* Initialize the specific angular momentum of cooling gas module.
*Code lines:* 42
*Contained by:* module `cooling_specific-angular-momenta`
*Modules used:* `galacticus_error` `input_parameters`

**file:** `cooling_specific-angular-momentum.constant_rotation.F90`
*Description:* Contains a module which calculates the specific angular momentum of cooling gas assuming a constant rotation velocity as a function of radius.
*Code lines:* 197

**module:** `cooling_specific-angular-momenta_constant_rotation`
*Description:* Calculates the specific angular momentum of cooling gas assuming a constant rotation velocity as a function of radius.
*Code lines:* 176
*Contained by:* file `cooling_specific-angular-momentum.constant_rotation.F90`
*Modules used:* `kind_numbers`
*Used by:* module `cooling_specific-angular-momenta` subroutine `galacticus_calculations_reset`

**subroutine:** `cooling_specific_am_constant_rotation_initialize`
*Description:* Initializes the “constant rotation” specific angular momentum of cooling gas module.
*Code lines:* 64
*Contained by:* module `cooling_specific-angular-momenta_constant_rotation`
*Modules used:* `input_parameters` `iso_varying_string`

**subroutine:** `cooling_specific_am_constant_rotation_reset`
*Description:* Reset the specific angular momentum of cooling gas calculation.
*Code lines:* 9
*Contained by:* module `cooling_specific-angular-momenta_constant_rotation`
*Modules used:* `galacticus_nodes`

**function:** `cooling_specific-angular-momentum_constant_rotation`
*Description:* Return the specific angular momentum of cooling gas in the constant rotation model.
*Code lines:* 67
*Contained by:* module `cooling_specific-angular-momenta_constant_rotation`
*Modules used:* `dark_matter_profiles` `galacticus_nodes` `hot_halo_mass_distributions` `numerical_constants_physical`

**file:** `cooling_specific-angular-momentum.mean.F90`
*Description:* Contains a module which calculates the specific angular momentum of cooling gas assuming all gas has the mean specific angular momentum of the hot gas halo.
*Code lines:* 65

**module:** `cooling_specific-angular-momenta_mean`
*Description:* Calculates the specific angular momentum of cooling gas assuming all gas has the mean specific angular momentum of the hot gas halo.
*Code lines:* 44
19.1. Program units

**Contained by:** file *cooling_specific-angular_momentum.mean.F90*

**Used by:** module *cooling_specific-angular_momenta*

**Subroutine:** *cooling_specific_am_mean_initialize*

*Description:* Initializes the “mean” specific angular momentum of cooling gas module.

*Code lines:* 16

**Modules used:** *array_utilities galacticus_error galacticus_nodes iso_varying_string*

**Function:** *cooling_specific-angular_momentum_mean*

*Description:* Return the specific angular momentum of cooling gas in the mean model.

*Code lines:* 12

**Modules used:** *galacticus_nodes*

**File:** *cooling.time_available.F90*

*Description:* Contains a module that implements calculations of the time available for cooling.

*Code lines:* 129

**Module:** *cooling_times_available*

*Code lines:* 109

**Contained by:** file *cooling.time_available.F90*

**Modules used:** *cooling_time_available_white_frenk cooling_times_available_halo-formation galacticus_nodes iso_varying_string*

**Used by:** function *cooling_radius_growth_rate-isothermal cooling_radius_simple*

**Function:** *cooling_time_available*

*Description:* Return the time available for cooling in *thisNode*.

*Code lines:* 12

**Contained by:** module *cooling_times_available*

**Interface:** *cooling_time_available_get_template*

*Code lines:* 5

**Contained by:** module *cooling_times_available*

**Function:** *cooling_time_available_get_template*

*Code lines:* 3

**Contained by:** interface *cooling_time_available_get_template*

**Function:** *cooling_time_available_increase_rate*

*Description:* Return the rate at which the time available for cooling increases in *thisNode*.

*Code lines:* 12

**Contained by:** module *cooling_times_available*
subroutine: cooling_time_available_initialize
  Description: Initializes the cooling time available module.
  Code lines: 42
  Contained by: module cooling_times_available
  Modules used: galacticus_error  input_parameters

file: cooling.time_available.White-Frenk.F90
  Description: Contains a module which implements the White and Frenk [1991] method for computing the time available for cooling in hot halos.
  Code lines: 105

module: cooling_time_available_white_frenk
  Description: Implements the White and Frenk [1991] method for computing the time available for cooling in hot halos.
  Code lines: 84
  Contained by: file cooling.time_available.White-Frenk.F90
  Used by: module cooling_times_available
  Modules used: galacticus_nodes

function: cooling_time_available_increase_rate_wf
  Description: Compute the rate of increase of the time available for cooling using the White and Frenk [1991] method. We return a rate of 1, even though technically it can depend on halo properties.
  Code lines: 10
  Contained by: module cooling_time_available_white_frenk
  Modules used: galacticus_nodes

function: cooling_time_available_wf
  Description: Compute the time available for cooling using the White and Frenk [1991] method. This is assumed to be equal to the dynamical timescale of the halo.
  Code lines: 26
  Contained by: module cooling_time_available_white_frenk
  Modules used: dark_matter_halo_scales  galacticus_nodes

subroutine: cooling_time_available_wf_initialize
  Description: Initialize the White and Frenk [1991] cooling time available module.
  Code lines: 28
  Contained by: module cooling_time_available_white_frenk
  Modules used: galacticus_error  input_parameters  iso_varying_string

file: cooling.time_available.halo_formation.F90
  Description: Contains a module which implements the Cole et al. [2000] method for computing the time available for cooling in hot halos.
  Code lines: 78

module: cooling_times_available_halo_formation
  Description: Implements the Cole et al. [2000] method for computing the time available for cooling in hot halos.
  Code lines: 57
19.1. Program units

*Contained by:* file `cooling.time_available.halo_formation.F90`

*Modules used:* `galacticus_nodes`

*Used by:* module `cooling_times_available`

**function:** `cooling_time_available_halo_formation`

*Description:* Compute the time available for cooling using the Cole et al. [2000] method. Specifically, the time available is assumed to be the time since the halo formation event.

*Code lines:* 13

*Contained by:* module `cooling_times_available_halo_formation`

**subroutine:** `cooling_time_available_halo_formation_initialize`

*Description:* Initialize the Cole et al. [2000] cooling time available module.

*Code lines:* 17

*Contained by:* module `cooling_times_available_halo_formation`

*Modules used:* `galacticus_error` `iso_varying_string`

**function:** `cooling_time_available_increase_rate_halo_formation`

*Description:* Compute the rate of increase of the time available for cooling using the Cole et al. [2000] method. We return a rate of 1.

*Code lines:* 9

*Contained by:* module `cooling_times_available_halo_formation`

**file:** `cosmology.functions.F90`

*Description:* Contains a module which provides an object that implements cosmological functions.

*Code lines:* 2648

**module:** `cosmology_functions`
Description: Provides an object that implements cosmological functions. An implementation of the cosmological functions class for cosmologies consisting of collisionless matter plus a cosmological constant. An implementation of the cosmological functions class for cosmologies consisting of collisionless matter and dark energy with an equation of state of the form: $P = \rho^w$ with $w(a) = w_0 + w_1(a - 1)$.

Code lines: 2628

Contained by: file cosmology.functions.F90

Modules used: cosmology_parameters fgs1
iso_c_binding iso_varying_string

Used by: program conditional_mass_function program tests_excursion_sets
function coldmodedefaultconstructor function simpledefaultconstructor
function coldmodedefaultconstructor function simpledefaultconstructor
subroutine cooling_rate_modifier_cutoff subroutine expansion_factor_at_formation
function dark_matter_halo_mass_- function dark_matter_halos_mass_-function
accretion_time_wechsler2002 loss_rate_vandenbosch
function dark_matter_halos_mass_-function dark_matter_halos_mass_-function
rate_vandenbosch loss_rate_vandenbosch_initialize
function
function virialdensitycontrastdefinitionmeandensityvirialdensitycontrastdefinitionmeandensitygrowth
function
function virialdensitycontrastdefinitionmeandensityvirialdensitycontrastdefinitionmeandensitygrowth
function
function duttonmaccio2014concentration function duttonmaccio2014concentration
function gao2008concentration function munozcuartas2011concentration
function nfw1996concentration function prada2011concentration
19.1. Program units

function gao2008concentration
function nfw1996concentration
function dark_matter_profile_scale
subroutine galacticus_output_analysis_mass_dpndnt_sz_dstrbtins
subroutine load_primus_mass_function
function mean_density_contrast_root

subroutine galacticus_merge_tree_output_filter_lightcone
subroutine galacticus_output_redshifts
subroutine galacticus_state_retrieve
function geometry_survey_distance_maximum_caputi_2011_ukidss_uds
function geometry_survey_distance_maximum_li_white_2009_sdss
subroutine geometry_survey_window_functions_martin_2010_alfalfa
function fileconstructor

module intergalactic_medium_state_internal_evolver
function intergalactic_medium_state_internal_odes
function simpleconstructor

subroutine merger_tree_conditional_mass_function
subroutine merger_tree_build_initialize
subroutine merger_tree_read_do
subroutine galacticusimport
subroutine galacticustreeindicesread
function importerunitconvert1d
function importerunitconvertscalar
function sussingtreweight
function sussingtreweight
subroutine galacticussubhalotrace
function galacticustreweight

function evolve_to_time
subroutine merger_tree_timestep_record_evolution

function munozcuertas2011concentration
function prada2011concentration
subroutine cosmology_conversion_factors
subroutine galacticus_output_analysis_mass_functions
subroutine make_output_group
subroutine galacticus_merge_tree_lightcone_geometry_initialize
subroutine galacticus_extra_output_halo_fourier_profile
subroutine output_times_initialize
subroutine galacticus_state_store
subroutine geometry_survey_window_functions_caputi_2011_ukidss_uds
subroutine geometry_survey_window_functions_li_white_2009_sdss
subroutine halo_mass_function_compute
subroutine filereaddata

subroutine filereaddata

function intergalacticmediumstateelectronscatteringtime
function intergalacticmediumstateelectronscatteringtabulate

function import_unit_convert1d
function import_unit_convert2d

function excursion_sets_maximum__sigma_test
subroutine merger_tree_build_cole2000_initialize
subroutine build_subhalo_mass_histories
subroutine phase_space_position_realize
subroutine galacticussubhalotrace
function galacticustreweight
function importerunitconvert2d
subroutine sussingtreweight
subroutine sussingtreweight
subroutine galacticussubhalotrace
subroutine merger_tree_smooth_accretion_do
subroutine merger_tree_timestep_history
subroutine merger_tree_timestep_simple
### Source Code Documentation

<table>
<thead>
<tr>
<th>Subroutine / Function Name</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>merger_tree_regrid_time</td>
<td>subroutine</td>
</tr>
<tr>
<td>merger_tree_write</td>
<td>subroutine</td>
</tr>
<tr>
<td>stellar_luminosities_-</td>
<td>subroutine</td>
</tr>
<tr>
<td>special_cases</td>
<td>subroutine</td>
</tr>
<tr>
<td>radiation_set_cmb</td>
<td>subroutine</td>
</tr>
<tr>
<td>radiation_igb_file_-</td>
<td>subroutine</td>
</tr>
<tr>
<td>initialize</td>
<td>subroutine</td>
</tr>
<tr>
<td>radiation_intergalactic_-</td>
<td>subroutine</td>
</tr>
<tr>
<td>background_internal_initi</td>
<td>subroutine</td>
</tr>
<tr>
<td>wetzelwhite2010timeuntilmerging</td>
<td>function</td>
</tr>
<tr>
<td>wetzelwhite2010timeuntilmerging</td>
<td>function</td>
</tr>
<tr>
<td>starformation_feedback_d</td>
<td>function</td>
</tr>
<tr>
<td>disk_-outflow_rate_halo_sc</td>
<td>function</td>
</tr>
<tr>
<td>module statistics_mass_f</td>
<td>module</td>
</tr>
<tr>
<td>function covariance</td>
<td>module</td>
</tr>
<tr>
<td>lss_integrand</td>
<td>function</td>
</tr>
<tr>
<td>volume_integrand</td>
<td>function</td>
</tr>
<tr>
<td>critical_overdensity_for_</td>
<td>function</td>
</tr>
<tr>
<td>collapse</td>
<td>function</td>
</tr>
<tr>
<td>time_ofCollapse</td>
<td>subroutine</td>
</tr>
<tr>
<td>excursion_sets_first_-</td>
<td>subroutine</td>
</tr>
<tr>
<td>crossing_rate_tabulate_farahi</td>
<td>function</td>
</tr>
<tr>
<td>linear_growth_factor</td>
<td>function</td>
</tr>
<tr>
<td>module linear_growth_simple</td>
<td>module</td>
</tr>
<tr>
<td>power_spectrum_nonlinear_</td>
<td>function</td>
</tr>
<tr>
<td>cosmicemu</td>
<td>function</td>
</tr>
<tr>
<td>peacockdodds1996</td>
<td>function</td>
</tr>
<tr>
<td>make_table</td>
<td>subroutine</td>
</tr>
<tr>
<td>bryannorman1998densityco</td>
<td>function</td>
</tr>
<tr>
<td>tratateofchange</td>
<td>function</td>
</tr>
<tr>
<td>fixeddensitycontrast</td>
<td>function</td>
</tr>
<tr>
<td>kitayamasuto1996densityco</td>
<td>function</td>
</tr>
<tr>
<td>tratateofchange</td>
<td>function</td>
</tr>
<tr>
<td>sphericalcollapsematterlambdadensitycontrast</td>
<td>function</td>
</tr>
<tr>
<td>kitayamasuto1996densityco</td>
<td>function</td>
</tr>
<tr>
<td>tragrateofchange</td>
<td>function</td>
</tr>
<tr>
<td>sphericalcollapsematterlambdadensitycontrast</td>
<td>function</td>
</tr>
<tr>
<td>kitayamasuto1996densityco</td>
<td>function</td>
</tr>
</tbody>
</table>

976
19.1. Program units

```plaintext
function kitayamasuto1996densitycontrastrateofchange
function fixeddensitycontrast
function fixeddensitycontrastrateofchange
function sphericalcollapsematterlambdadensitycontrast
program test_nfw96_concentration_dark_energy
program test_prada2011_concentration
program test_zhao2009_dark_energy
program tests_comoving_distance
program tests_halo_mass_function_tinker
program tests_linear_growth_kosmological_constant
program tests_linear_growth_dark_energy
program tests_linear_growth_open
program tests_spherical_collapse_kosmological_constant
program tests_spherical_collapse_dark_energy_eds
program tests_spherical_collapse_dark_energy_open
program tests_spherical_collapse_flat
program tests_spherical_collapse_open

interface: cosmologyfunctions
Code lines: 3
Contained by: module cosmology_functions

function: cosmologyfunctions_c
Code lines: 9
Contained by: module cosmology_functions

type: cosmologyfunctionsclass
Code lines: 217
Contained by: module cosmology_functions

function: cosmologyfunctionscomovingvolumeelementredshift
Description: Returns the differential comoving volume element \( \frac{dV}{dz} = r_c^2(t) c H^{-1}(t) \) (where \( r_c \) is the comoving distance to time \( t \) and \( H(t) \) is the Hubble parameter at that time) for unit solid angle at the specified time.
Code lines: 8
Contained by: module cosmology_functions
Modules used: numerical_constants_physical

function: cosmologyfunctionscomovingvolumeelementtime
Description: Returns the differential comoving volume element \( \frac{dV}{dt} = r_c^2(t) c a(t) \) (where \( r_c \) is the comoving distance to time \( t \) and \( a(t) \) is the expansion at that time) for unit solid angle at the specified time.
Code lines: 9
Contained by: module cosmology_functions
Modules used: numerical_constants_astronomical numerical_constants_physical
```

977
function: cosmologyfunctionsconstructordefault
  Description: Return a pointer to the default cosmologyFunctions object.
  Code lines: 8
  Contained by: module cosmology_functions

function: cosmologyfunctionsconstructornamed
  Description: Return a pointer to a newly created cosmologyFunctions object of the specified type.
  Code lines: 30
  Contained by: module cosmology_functions
  Modules used: galacticus_error iso_varying_string

function: cosmologyfunctionscosmictimeisvalidnull
  Description: Returns true if the given cosmic time is valid one for this cosmology.
  Code lines: 8
  Contained by: module cosmology_functions
  Modules used: galacticus_error

function: cosmologyfunctionscosmictimenull
  Description: Return the cosmological age at the given expansion factor.
  Code lines: 9
  Contained by: module cosmology_functions
  Modules used: galacticus_error

subroutine: cosmologyfunctionsdensityscalingearlytimenull
  Description: Compute the scaling of density with expansion factor at early times in the universe.
  Code lines: 10
  Contained by: module cosmology_functions
  Modules used: galacticus_error

subroutine: cosmologyfunctionsdestructor_c
  Code lines: 8
  Contained by: module cosmology_functions

function: cosmologyfunctionsdistanceangularnull
  Description: Return the angular diameter distance to the given cosmic time.
  Code lines: 8
  Contained by: module cosmology_functions
  Modules used: galacticus_error

function: cosmologyfunctionsdistancecomovingconvertnull
  Description: Convert between different measures of comoving distance.
  Code lines: 9
  Contained by: module cosmology_functions
  Modules used: galacticus_error

function: cosmologyfunctionsdistancecomovingnull
  Description: Return the comoving distance to the given cosmic time.
  Code lines: 8
19.1. Program units

**Contained by:** module `cosmology_functions`
**Modules used:** `galacticus_error`

**function:** `cosmologyfunctionsdistanceluminositynull`
**Description:** Return the luminosity distance to the given cosmic time.
**Code lines:** 8
**Contained by:** module `cosmology_functions`
**Modules used:** `galacticus_error`

**function:** `cosmologyfunctionsdominationepochmatternull`
**Description:** Compute the epoch at which matter dominates over other forms of energy by a given factor.
**Code lines:** 8
**Contained by:** module `cosmology_functions`
**Modules used:** `galacticus_error`

**subroutine:** `cosmologyfunctionsdostructuretrieve`
**Description:** Retrieve the state from file.
**Code lines:** 10
**Contained by:** module `cosmology_functions`

**subroutine:** `cosmologyfunctionsdostructurestore`
**Description:** Store the state to file.
**Code lines:** 10
**Contained by:** module `cosmology_functions`

**function:** `cosmologyfunctionsepochtime`
**Description:** Convenience function that returns the time corresponding to an epoch specified by time or expansion factor.
**Code lines:** 17
**Contained by:** module `cosmology_functions`
**Modules used:** `galacticus_error`

**function:** `cosmologyfunctionsequalityepochmattercurvaturenull`
**Description:** Return the epoch of matter-curvature magnitude equality (either expansion factor or cosmic time).
**Code lines:** 8
**Contained by:** module `cosmology_functions`
**Modules used:** `galacticus_error`

**function:** `cosmologyfunctionsequalityepochmatterdarkenergynull`
**Description:** Return the epoch of matter-dark energy magnitude equality (either expansion factor or cosmic time).
**Code lines:** 8
**Contained by:** module `cosmology_functions`
**Modules used:** `galacticus_error`

**function:** `cosmologyfunctionsequationofstatedarkenergynull`
**Description:** HASH(0x1989a30)
**Code lines:** 8
19. Source Code Documentation

**Contained by:**  module `cosmology_functions`

**Modules used:**  `galacticus_error`

**function:**  `cosmologyfunctionsexpansionfactorfromredshift`

**Description:**  Returns expansion factor given a redshift.

**Code lines:**  7

**Contained by:**  module `cosmology_functions`

**function:**  `cosmologyfunctionsexpansionfactorisvalidnull`

**Description:**  Returns true if the given expansion factor is valid one for this cosmology.

**Code lines:**  8

**Contained by:**  module `cosmology_functions`

**Modules used:**  `galacticus_error`

**function:**  `cosmologyfunctionsexpansionfactornull`

**Description:**  Returns the expansion factor at cosmological time `time`.

**Code lines:**  8

**Contained by:**  module `cosmology_functions`

**Modules used:**  `galacticus_error`

**function:**  `cosmologyfunctionsexpansionratenull`

**Description:**  Returns the cosmological expansion rate, $\dot{a}/a$ at expansion factor `expansionFactor`.

**Code lines:**  8

**Contained by:**  module `cosmology_functions`

**Modules used:**  `galacticus_error`

**function:**  `cosmologyfunctionsexponentdarkenergynull`

**Description:**  HASH(0x198caa8)

**Code lines:**  8

**Contained by:**  module `cosmology_functions`

**Modules used:**  `galacticus_error`

**function:**  `cosmologyfunctionsinitialize`

**Description:**  Initialize the default `cosmologyFunctions` object.

**Code lines:**  47

**subroutine:**  `cosmologyfunctionsinitialize`

**Description:**  Initialize the default `cosmologyFunctions` object.

**Code lines:**  47
19.1. Program units

Contained by: module cosmology_functions
Modules used: galacticus_error input_parameters

function: cosmologyfunctionsisfinalizable
Description: Return true if this object can be finalized.
Code lines: 7
Contained by: module cosmology_functions

interface: cosmologyfunctionsmatterdarkenergy
Description: Constructors for the matter plus dark energy cosmological functions class.
Code lines: 4
Contained by: module cosmology_functions

interface: cosmologyfunctions Matterlambda
Description: Constructors for the matter plus cosmological constant cosmological functions class.
Code lines: 4
Contained by: module cosmology_functions

function: cosmologyfunctionsomegadarkenergypeochalnull
Description: Return the dark energy density parameter at expansion factor expansionFactor.
Code lines: 9
Contained by: module cosmology_functions
Modules used: galacticus_error

function: cosmologyfunctionsomegamatterepochalnull
Description: Return the matter density parameter at expansion factor expansionFactor.
Code lines: 9
Contained by: module cosmology_functions
Modules used: galacticus_error

function: cosmologyfunctionsomegamatterrateofchangenull
Description: Return the rate of change of the matter density parameter at expansion factor expansionFactor.
Code lines: 9
Contained by: module cosmology_functions
Modules used: galacticus_error

function: cosmologyfunctionsredshiftfromexpansionfactor
Description: Returns redshift for a given expansion factor.
Code lines: 7
Contained by: module cosmology_functions

subroutine: cosmologyfunctionsstaterestore
Description: Restore the state of the object to file.
Code lines: 9
Contained by: module cosmology_functions
Modules used: fgsl
subroutine: cosmologyfunctionsstatestore
Description: Store the state of the object to file.
Code lines: 9
Contained by: module cosmology_functions
Modules used: gsl

function: cosmologyfunctionstemperaturecmbepochalnull
Description: Return the temperature of the cosmic microwave background at expansionFactor.
Code lines: 9
Contained by: module cosmology_functions
Modules used: galacticus_error

function: cosmologyfunctionstimeatdistancecomovingnull
Description: Return the cosmic time corresponding to the given comovingDistance.
Code lines: 8
Contained by: module cosmology_functions
Modules used: galacticus_error

type: cosmologyfunctionswrapper
Code lines: 2
Contained by: module cosmology_functions

function: expansionfactor_c
Code lines: 8
Contained by: module cosmology_functions

function: matterdarkenergyagetableodes
Description: System of differential equations to solve for expansion factor vs. age.
Code lines: 15
Contained by: module cosmology_functions
Modules used: iso_c_binding

function: matterdarkenergycosmictime
Description: Return the cosmological matter density in units of the critical density at the present day.
Code lines: 38
Contained by: module cosmology_functions
Modules used: galacticus_error numerical_interpolation

function: matterdarkenergycosmictime
Description: Default constructor for the matter plus dark energy cosmological functions class.
19.1. Program units

Code lines: 45
Contained by: module cosmolology_functions
Modules used: cosmology_parameters input_parameters

function: matterdarkenergydistancecomoving
Description: Returns the comoving distance to cosmological time \( t \).
Code lines: 9
Contained by: module cosmolology_functions
Modules used: galacticus_error

function: matterdarkenergydistancecomovingconvert
Description: Convert between different measures of distance.
Code lines: 10
Contained by: module cosmolology_functions
Modules used: galacticus_error

function: matterdarkenergydomination
Description: Function used in root finding when seeking epoch at which matter dominates over dark energy.
Code lines: 7
Contained by: module cosmolology_functions

function: matterdarkenergydominationepochmatter
Code lines: 59
Contained by: module cosmolology_functions
Modules used: cosmology_functions_parameters root_finder

function: matterdarkenergyequalityepochmatterdarkenergy
Description: Return the epoch of matter-dark energy magnitude equality (either expansion factor or cosmic time).
Code lines: 54
Contained by: module cosmolology_functions
Modules used: cosmology_functions_parameters cosmology_parameters root_finder

function: matterdarkenergyequationofstatedarkenergy
Description: Return the dark energy equation of state.
Code lines: 18
Contained by: module cosmolology_functions
Modules used: galacticus_error

function: matterdarkenergyexpansionfactorchange
Description: Compute the expansion factor at time \( t_{\text{End}} \) given an initial value \( \text{expansionFactor}_{\text{Start}} \) at time \( t_{\text{Start}} \).
Code lines: 23
Contained by: module cosmolology_functions
Modules used: iso_c_binding ode_solver

function: matterdarkenergyexponentdarkenergy
Description: Return the dark energy exponent.
Code lines: 22
Contained by: module `cosmology_functions`
Modules used: `galacticus_error`

**function:** `matterdarkenergyexponentdarkenergyderivative`
Description: Return the derivative of the dark energy exponent with respect to expansion factor.
Code lines: 22
Contained by: module `cosmology_functions`
Modules used: `galacticus_error`

**function:** `matterdarkenergyhubbleparameterepochal`
Description: Returns the Hubble parameter at the request cosmological time, time, or expansion factor, expansionFactor.
Code lines: 37
Contained by: module `cosmology_functions`
Modules used: `galacticus_error`

**function:** `matterdarkenergyhubbleparameterrateofchange`
Description: Returns the rate of change of the Hubble parameter at the requested cosmological time, time, or expansion factor, expansionFactor.
Code lines: 26
Contained by: module `cosmology_functions`
Modules used: `galacticus_error`

**subroutine:** `matterdarkenergymakeexpansionfactortable`
Description: Builds a table of expansion factor vs. time for dark energy universes.
Code lines: 127
Contained by: module `cosmology_functions`
Modules used: `cosmology_parameters iso_c_binding memory_management numerical_interpolation numerical_ranges`

**function:** `matterdarkenergyomegadarkenergyepochal`
Description: Return the dark energy density parameter at expansion factor expansionFactor.
Code lines: 25
Contained by: module `cosmology_functions`
Modules used: `galacticus_error`

**subroutine:** `matterdarkenergytargetself`
Description: Set a module-scope pointer to the current dark energy cosmology functions object.
Code lines: 7
Contained by: module `cosmology_functions`

**function:** `matterdarkenergytimeatdistancecomoving`
Description: Returns the cosmological time corresponding to given comovingDistance.
Code lines: 9
Contained by: module `cosmology_functions`
Modules used: `galacticus_error`
function: matterlambdaagetableodes
Description: System of differential equations to solve for expansion factor vs. age.
Code lines: 11
Contained by: module cosmology_functions
Modules used: iso_c_binding

function: matterlambdacollapseodes
Description: System of differential equations to solve for age vs. expansion factor.
Code lines: 13
Contained by: module cosmology_functions
Modules used: iso_c_binding

function: matterlambdacomovingdistanceintegrand
Description: Integrand function used in computing the comoving distance.
Code lines: 12
Contained by: module cosmology_functions
Modules used: iso_c_binding numerical_constants_astronomical numerical_constants_physical

function: matterlambdaconstructor
Description: Constructor for the matter plus cosmological constant cosmological functions class.
Code lines: 84
Contained by: module cosmology_functions
Modules used: cosmology_parameters iso_c_binding iso_varying_string numerical_comparison ode_solver

function: matterlambdacosmictime
Description: Return the cosmological matter density in units of the critical density at the present day.
Code lines: 42
Contained by: module cosmology_functions
Modules used: galacticus_error numerical_interpolation

function: matterlambdacosmictimeisvalid
Description: Checks that the time falls within allowed ranges.
Code lines: 8
Contained by: module cosmology_functions

function: matterlambdadefaultconstructor
Description: Default constructor for the matter plus cosmological constant cosmological functions class.
Code lines: 12
Contained by: module cosmology_functions
Modules used: cosmology_parameters

subroutine: matterlambdadensityscalingearlytime
Code lines: 15
Contained by: module cosmology_functions
Modules used: cosmology_parameters
**subroutine: matterlambdadestructor**
*Description:* Default constructor for the matter plus cosmological constant cosmological functions class.
*Code lines:* 16
*Contained by:* module cosmology_functions
*Modules used:* numerical_interpolation

**function: matterlambdadistanceangular**
*Description:* Returns the angular diameter distance to cosmological time \( t \).
*Code lines:* 11
*Contained by:* module cosmology_functions
*Modules used:* galacticus_error numerical_interpolation

**function: matterlambdadistancecomoving**
*Description:* Returns the comoving distance to cosmological time \( t \).
*Code lines:* 24
*Contained by:* module cosmology_functions
*Modules used:* galacticus_error numerical_interpolation

**function: matterlambdadistancecomovingconvert**
*Description:* Convert between different measures of distance.
*Code lines:* 37
*Contained by:* module cosmology_functions
*Modules used:* cosmology_functions_options galacticus_error numerical_interpolation

**function: matterlambdadistanceluminosity**
*Description:* Returns the luminosity distance to cosmological time \( t \).
*Code lines:* 11
*Contained by:* module cosmology_functions
*Modules used:* galacticus_error numerical_interpolation

**function: matterlambdaordinationepochmatter**
*Code lines:* 29
*Contained by:* module cosmology_functions
*Modules used:* cosmology_functions_parameters cosmology_parameters

**function: matterlambdadequalityepochmattercurvature**
*Description:* Return the epoch of matter-curvature magnitude equality (either expansion factor or cosmic time).
*Code lines:* 18
*Contained by:* module cosmology_functions
*Modules used:* cosmology_functions_parameters cosmology_parameters

**function: matterlambdadequalityepochmatterdarkenergy**
*Description:* Return the epoch of matter-dark energy magnitude equality (either expansion factor or cosmic time).
*Code lines:* 17
*Contained by:* module cosmology_functions
*Modules used:* cosmology_functions_parameters cosmology_parameters
function: matterlambdaequationofstatedarkenergy
Description: Return the dark energy equation of state.
Code lines: 8
Contained by: module cosmology_functions

function: matterlambdaexpansionfactor
Description: Returns the expansion factor at cosmological time time.
Code lines: 39
Contained by: module cosmology_functions
Modules used: galacticus_error numerical_interpolation

function: matterlambdaexpansionfactorisvalid
Description: Checks that the expansion factor falls within allowed ranges.
Code lines: 8
Contained by: module cosmology_functions

function: matterlambdaexpansionrate
Description: Returns the cosmological expansion rate, $\dot{a}/a$ at expansion factor expansionFactor.
Code lines: 10
Contained by: module cosmology_functions
Modules used: cosmology_parameters

function: matterlambdaexponentdarkenergy
Description: Return the dark energy equation of state.
Code lines: 8
Contained by: module cosmology_functions

function: matterlambdahubbleparameterepochal
Description: Returns the Hubble parameter at the request cosmological time, time, or expansion factor, expansionFactor.
Code lines: 38
Contained by: module cosmology_functions
Modules used: cosmology_parameters galacticus_error

function: matterlambdahubbleparameterrateofchange
Description: Returns the rate of change of the Hubble parameter at the request cosmological time, time, or expansion factor, expansionFactor.
Code lines: 26
Contained by: module cosmology_functions
Modules used: cosmology_parameters galacticus_error

subroutine: matterlambdamadeakedistancetable
Description: Builds a table of distance vs. time.
Code lines: 54
Contained by: module cosmology_functions
Modules used: iso_c_binding memory_management numerical_integration numerical_interpolation
numerical_ranges

subroutine: matterlambdamakeexpansionfactortable
Description: Builds a table of expansion factor vs. time.
Code lines: 104
Contained by: module cosmology_functions
Modules used: cosmology_parameters
iso_c_binding
numerical_interpolation
ode_solver

function: matterlambdaomegadarkenergyepochal
Description: Return the dark energy density parameter at expansion factor expansionFactor.
Code lines: 25
Contained by: module cosmology_functions
Modules used: galacticus_error

function: matterlambdaomegammatterepochal
Description: Return the matter density parameter at expansion factor expansionFactor.
Code lines: 27
Contained by: module cosmology_functions
Modules used: galacticus_error

function: matterlambdaomegammatterrateofchange
Description: Return the rate of change of the matter density parameter at expansion factor expansionFactor.
Code lines: 18
Contained by: module cosmology_functions
Modules used: galacticus_error

subroutine: matterlambdastaterestore
Description: Retrieve the tabulation state from the file.
Code lines: 29
Contained by: module cosmology_functions
Modules used: memory_management

subroutine: matterlambdastatestore
Description: Write the tablulation state to file.
Code lines: 14
Contained by: module cosmology_functions

function: matterlambdatemperaturecmbepochal
Description: Return the temperature of the CMB at expansion factor expansionFactor.
Code lines: 25
Contained by: module cosmology_functions
Modules used: galacticus_error

function: matterlambdatimeatdistancecomoving
Description: Returns the cosmological time corresponding to given comovingDistance.
19.1. Program units

Code lines: 28
Contained by: module cosmology_functions
Modules used: galacticus_error numerical_interpolation

file: cosmology_functions.matter_dark_energy.F90
Description: An implementation of the cosmological functions class for cosmologies consisting of collisionless matter and dark energy with an equation of state of the form: \( P = \rho^w \) with \( w(a) = w_0 + w_1 a (1 - a) \).
Code lines: 700
Modules used: cosmology_parameters fgsl

interface: cosmology_functions.matter_dark_energy
Description: Constructors for the matter plus dark energy cosmological functions class.
Code lines: 4
Contained by: file cosmology_functions.matter_dark_energy.F90

function: matterdarkenergygetableodes
Description: System of differential equations to solve for expansion factor vs. age.
Code lines: 15
Contained by: file cosmology_functions.matter_dark_energy.F90
Modules used: iso_c_binding

function: matterdarkenergyconstructor
Description: Constructor for the matter plus dark energy cosmological functions class.
Code lines: 20
Contained by: file cosmology_functions.matter_dark_energy.F90
Modules used: cosmology_parameters iso_c_binding iso_varying_string numerical_comparison ode_solver

function: matterdarkenergycosmictime
Description: Return the cosmological matter density in units of the critical density at the present day.
Code lines: 38
Contained by: file cosmology_functions.matter_dark_energy.F90
Modules used: galacticus_error numerical_interpolation

function: matterdarkenergydefaultconstructor
Description: Default constructor for the matter plus dark energy cosmological functions class.
Code lines: 45
Contained by: file cosmology_functions.matter_dark_energy.F90
Modules used: cosmology_parameters input_parameters

function: matterdarkenergydistancecomoving
Description: Returns the comoving distance to cosmological time \( t \).
Code lines: 9
Contained by: file cosmology_functions.matter_dark_energy.F90
Modules used: galacticus_error
function: matterdarkenergydistancecomovingconvert
Description: Convert between different measures of distance.
Code lines: 10
Contained by: file cosmology.functions.matter_dark_energy.F90
Modules used: galacticus_error

function: matterdarkenergydomination
Description: Function used in root finding when seeking epoch at which matter dominates over dark energy.
Code lines: 7
Contained by: file cosmology.functions.matter_dark_energy.F90

function: matterdarkenergydominationepochmatter
Code lines: 59
Contained by: file cosmology.functions.matter_dark_energy.F90
Modules used: cosmology_functions_parameters root_finder

function: matterdarkenergyequalityepochmatterdarkenergy
Description: Return the epoch of matter-dark energy magnitude equality (either expansion factor or cosmic time).
Code lines: 54
Contained by: file cosmology.functions.matter_dark_energy.F90
Modules used: cosmology_functions_parameters cosmology_parameters root_finder

function: matterdarkenergequationofstatedarkenergy
Description: Return the dark energy equation of state.
Code lines: 18
Contained by: file cosmology.functions.matter_dark_energy.F90
Modules used: galacticus_error

function: matterdarkenergeexpansionfactorchange
Description: Compute the expansion factor at time timeEnd given an initial value expansionFactorStart at time timeStart.
Code lines: 23
Contained by: file cosmology.functions.matter_dark_energy.F90
Modules used: iso_c_binding ode_solver

function: matterdarkenergyexponentdarkenergy
Description: Return the dark energy exponent.
Code lines: 22
Contained by: file cosmology.functions.matter_dark_energy.F90
Modules used: galacticus_error

function: matterdarkenergyexponentdarkenergysderivative
Description: Return the derivative of the dark energy exponent with respect to expansion factor.
Code lines: 22
Contained by: file cosmology.functions.matter_dark_energy.F90
19.1. Program units

Modules used: galacticus_error

function: matterdarkenergyhubbleparameterepochal
Description: Returns the Hubble parameter at the request cosmological time, time, or expansion factor, expansionFactor.
Code lines: 37
Contained by: file cosmology.functions.matter_dark_energy.F90
Modules used: galacticus_error

function: matterdarkenergyhubbleparameterrateofchange
Description: Returns the rate of change of the Hubble parameter at the requested cosmological time, time, or expansion factor, expansionFactor.
Code lines: 26
Contained by: file cosmology.functions.matter_dark_energy.F90
Modules used: galacticus_error

subroutine: matterdarkenergymakeexpansionfactortable
Description: Builds a table of expansion factor vs. time for dark energy universes.
Code lines: 127
Contained by: file cosmology.functions.matter_dark_energy.F90
Modules used: cosmology_parameters iso_c_binding memory_management numerical_interpolation numerical_ranges

function: matterdarkenergyomegadarkenergymomegadarkenergyepochal
Description: Return the dark energy density parameter at expansion factor expansionFactor.
Code lines: 25
Contained by: file cosmology.functions.matter_dark_energy.F90
Modules used: galacticus_error

subroutine: matterdarkenergytargetself
Description: Set a module-scope pointer to the current dark energy cosmology functions object.
Code lines: 7
Contained by: file cosmology.functions.matter_dark_energy.F90

function: matterdarkenergymtimeatdistancecomoving
Description: Returns the cosmological time corresponding to given comovingDistance.
Code lines: 9
Contained by: file cosmology.functions.matter_dark_energy.F90
Modules used: galacticus_error

file: cosmology.functions.matter_lambda.F90
Description: An implementation of the cosmological functions class for cosmologies consisting of collisionless matter plus a cosmological constant.
Code lines: 997
Modules used: cosmology_parameters fgs1

interface: cosmologyfunctions.matterlambda
Description: Constructors for the matter plus cosmological constant cosmological functions class.
19. Source Code Documentation

function: matterlambdaagetableodes
Description: System of differential equations to solve for expansion factor vs. age.
Code lines: 11
Contained by: file cosmology.functions.matter_lambda.F90
Modules used: iso_c_binding

function: matterlambdacollapseodes
Description: System of differential equations to solve for age vs. expansion factor.
Code lines: 13
Contained by: file cosmology.functions.matter_lambda.F90
Modules used: iso_c_binding

function: matterlambdacomovingdistanceintegrand
Description: Integrand function used in computing the comoving distance.
Code lines: 12
Contained by: file cosmology.functions.matter_lambda.F90
Modules used: iso_c_binding numerical_constants_astronomical numerical_constants_physical

function: matterlambdacometrical
Description: Constructor for the matter plus cosmological constant cosmological functions class.
Code lines: 84
Contained by: file cosmology.functions.matter_lambda.F90
Modules used: cosmology_parameters iso_c_binding iso_varying_string numerical_comparison ode_solver

function: matterlambdacosmictime
Description: Return the cosmological matter density in units of the critical density at the present day.
Code lines: 42
Contained by: file cosmology.functions.matter_lambda.F90
Modules used: galacticus_error numerical_interpolation

function: matterlambdacosmictimeisvalid
Description: Checks that the time falls within allowed ranges.
Code lines: 8
Contained by: file cosmology.functions.matter_lambda.F90

function: matterlambdadefaultconstructor
Description: Default constructor for the matter plus cosmological constant cosmological functions class.
Code lines: 12
Contained by: file cosmology.functions.matter_lambda.F90
Modules used: cosmology_parameters

subroutine: matterlambdadensityscalingearlytime
Code lines: 15
19.1. Program units

**Contained by:** file `cosmology.functions.matter_lambda.F90`  
**Modules used:** `cosmology_parameters`

**subroutine:** `matterlambdadestructor`  
**Description:** Default constructor for the matter plus cosmological constant cosmological functions class.  
**Code lines:** 16  
**Contained by:** file `cosmology.functions.matter_lambda.F90`  
**Modules used:** `numerical_interpolation`

**function:** `matterlambdadistanceangular`  
**Description:** Returns the angular diameter distance to cosmological time \( t \).  
**Code lines:** 11  
**Contained by:** file `cosmology.functions.matter_lambda.F90`  
**Modules used:** `galacticus_error` `numerical_interpolation`

**function:** `matterlambdadistancecomoving`  
**Description:** Returns the comoving distance to cosmological time \( t \).  
**Code lines:** 24  
**Contained by:** file `cosmology.functions.matter_lambda.F90`  
**Modules used:** `galacticus_error` `numerical_interpolation`

**function:** `matterlambdadistancecomovingconvert`  
**Description:** Convert between different measures of distance.  
**Code lines:** 37  
**Contained by:** file `cosmology.functions.matter_lambda.F90`  
**Modules used:** `cosmology_functions_options` `galacticus_error` `numerical_interpolation`

**function:** `matterlambdadistanceluminosity`  
**Description:** Returns the luminosity distance to cosmological time \( t \).  
**Code lines:** 11  
**Contained by:** file `cosmology.functions.matter_lambda.F90`  
**Modules used:** `galacticus_error` `numerical_interpolation`

**function:** `matterlambdadominationepochmatter`  
**Code lines:** 29  
**Contained by:** file `cosmology.functions.matter_lambda.F90`  
**Modules used:** `cosmology_functions_parameters` `cosmology_parameters`

**function:** `matterlambdaequalityepochmattercurvature`  
**Description:** Return the epoch of matter-curvature magnitude equality (either expansion factor or cosmic time).  
**Code lines:** 18  
**Contained by:** file `cosmology.functions.matter_lambda.F90`  
**Modules used:** `cosmology_functions_parameters` `cosmology_parameters`

**function:** `matterlambdaequalityepochmatterdarkenergy`  
**Description:** Return the epoch of matter-dark energy magnitude equality (either expansion factor or cosmic time).  
**Code lines:** 17
19. Source Code Documentation

function: matterlambdaequationofstatedarkenergy
Description: Return the dark energy equation of state.
Code lines: 8
-contained by: file cosine.functions.matter_lambda.F90

Modules used: cosmology_functions_parameters cosmology_parameters

function: matterlambdaexpansionfactor
Description: Returns the expansion factor at cosmological time time.
Code lines: 39
-contained by: file cosine.functions.matter_lambda.F90

Modules used: galacticus_error numerical_interpolation

function: matterlambdaexpansionfactorisvalid
Description: Checks that the expansion factor falls within allowed ranges.
Code lines: 8
-contained by: file cosine.functions.matter_lambda.F90

function: matterlambdaexpansionrate
Description: Returns the cosmological expansion rate, \( \dot{a}/a \) at expansion factor expansionFactor.
Code lines: 10
-contained by: file cosine.functions.matter_lambda.F90

Modules used: cosmology_parameters

function: matterlambdaexponentdarkenergy
Description: Return the dark energy equation of state.
Code lines: 8
-contained by: file cosine.functions.matter_lambda.F90

function: matterlambdahubbleparameterepochal
Description: Returns the Hubble parameter at the request cosmological time, time, or expansion factor, expansionFactor.
Code lines: 38
-contained by: file cosine.functions.matter_lambda.F90

Modules used: cosmology_parameters galacticus_error

function: matterlambdahubbleparameterrateofchange
Description: Returns the rate of change of the Hubble parameter at the request cosmological time, time, or expansion factor, expansionFactor.
Code lines: 26
-contained by: file cosine.functions.matter_lambda.F90

Modules used: cosmology_parameters galacticus_error

subroutine: matterlambdamakedistancetable
Description: Builds a table of distance vs. time.
Code lines: 54
-contained by: file cosine.functions.matter_lambda.F90

Modules used: iso_c_binding memory_management
subroutine: matterlambdamakeexpansionfactortable
Description: Builds a table of expansion factor vs. time.
Code lines: 104
Contained by: file cosmology.functions.matter_lambda.F90
Modules used: cosmology_parameters fgsl
iso_c_binding memory_management
numerical_interpolation numerical_ranges
ode_solver

function: matterlambdaomegadarkenergyepochal
Description: Return the dark energy density parameter at expansion factor expansionFactor.
Code lines: 25
Contained by: file cosmology.functions.matter_lambda.F90
Modules used: galacticus_error

function: matterlambdaomegammatterepochal
Description: Return the matter density parameter at expansion factor expansionFactor.
Code lines: 27
Contained by: file cosmology.functions.matter_lambda.F90
Modules used: galacticus_error

function: matterlambdaomegammatterrateofchange
Description: Return the rate of change of the matter density parameter at expansion factor expansionFactor.
Code lines: 18
Contained by: file cosmology.functions.matter_lambda.F90
Modules used: galacticus_error

subroutine: matterlambdastaterestore
Description: Retrieve the tabulation state from the file.
Code lines: 29
Contained by: file cosmology.functions.matter_lambda.F90
Modules used: memory_management

subroutine: matterlambdastatestore
Description: Write the tabulation state to file.
Code lines: 14
Contained by: file cosmology.functions.matter_lambda.F90

function: matterlambdatemperaturecmbepochal
Description: Return the temperature of the CMB at expansion factor expansionFactor.
Code lines: 25
Contained by: file cosmology.functions.matter_lambda.F90
Modules used: galacticus_error

function: matterlambdatimeatdistancecomoving
Description: Returns the cosmological time corresponding to given comovingDistance.
Code lines: 28
Contained by: file cosmology.functions.matter_lambda.F90
Modules used: galacticus_error numerical_interpolation

file: cosmology.functions.options.F90
Description: Contains a module which provides options for cosmological functions.
Code lines: 27

module: cosmology_functions_options
Description: Provides options for cosmological functions.
Code lines: 7
Contained by: file cosmology.functions.options.F90
Used by: function matterlambdadistancecomovingconvert
function geometry_survey_distance_-maximum_caputi_2011_ukidss_uds
program tests_comoving_distance

file: cosmology.functions.parameters.F90
Description: Contains a module which provides an object that implements cosmological parameters.
Code lines: 576

module: cosmology_functions_parameters
Description: Defines option parameters for cosmology functions.
Code lines: 9
Contained by: file cosmology.functions.parameters.F90
Used by: function matterdarkenergiedominationepochmatter
function matterdarkenergyequalityepochmatterdarkenergy
function matterlambdadominationepochmatter
function matterlambdaequalityepochmattercurvature
function matterlambdaequalityepochmatterdarkenergymatterlambdadominationepochmatter
function matterdarkenergiedqualityepochmatterdarkenergymatterlambdadominationepochmatter
function matterdarkenergiedequalityepochmatterdarkenergymatterlambdadominationepochmatter
function matterdarkenergiedqualityepochmatterdarkenergymatterlambdaequalityepochmatterdarkenergy

file: cosmology.parameters.F90
Description: Contains a module which provides an object that implements cosmological parameters.
Code lines: 576

module: cosmology_parameters
Description: Provides an object that implements cosmological parameters. A simple implementation of the cosmological parameters class.
Code lines: 556
Contained by: file cosmology.parameters.F90
Modules used: iso_varying_string
Used by: program tests_excursion_sets function coldmodechemicalmasses
function coldmodecoldmodefraction
function simpleaccretedmass
function simplechemicalmasses
function simplefailedaccretionrate
function coldmodechemicalmasses
function simpleaccretedmass
function simplechemicalmasses
function simplefailedaccretionrate
function matterdarkenergyconstructor

function matterdarkenergyequalityepochmatterdarkenergymakeexpansionfactortable
function matterlambdaconstructor
subroutine
matterlambdadenstydscaleingearlytime
function
matterlambdaequalityepochmattercurvature
function matterlambdaexpansionrate

function matterlambdahubbleparameterepochal
subroutine
matterlambdahubbleparameterrateofchange
file cosmology.functions.matter_dark_energy.F90
function
matterdarkenergyconstructor

function matterlambdadenstydscaleingearlytime
function
matterlambdaequalityepochmatterdarkenergymakeexpansionfactortable

function matterlambdahubbleparameterrateofchange
function virialdensitycontrastdefinitionmeandensity
function


function diemerkravtsov2014concentration
diemerkravtsov2014concentration

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function

function
function dark_matter_profile_enclosed_mass_task
subroutine galactic_structure_radii_initial_adiabatic_compute_factors
subroutine galacticus_output_analysis_mass_functions
subroutine galacticus_output_tree_density_contrast_initialize
function geometry_survey_distance_maximum_martin_2010_alfalfa
function growingradius
function intergalacticmediumstateelectrons_scattering_integrand
function recfastDefaultConstructor
function recfastDefaultConstructor
subroutine intergalactic_medium_state_internal_initialize
function intergalactic_medium_state_internal_update
subroutine galacticusopen
function importerunitconvert2d
subroutine sussingopen
subroutine galacticusopen
subroutine merger_tree_write
subroutine node_component_hot_halo_cold_mode_node_merger
subroutine node_component_hot_halo_standard_node_merger
subroutine node_component_hot_halo_standard_rate_compute
subroutine power_spectrum_compute
function variance_integral
subroutine critical_overdensity_mass_scaling_wdm_initialize
function halo_mass_function_differential_press_schechter
function halo_mass_function_differential_tinker2008
subroutine initialize_cosmological_mass_variance
subroutine galactic_structure_radii_solve_adiabatic
subroutine galacticus_output_analysis_mass_dpmdnt_sz_dstrbtins
subroutine load_primus_mass_function
subroutine galacticus_merger_tree_lightcone_geometry_init
function geometry_survey_window_functions_martin_2010_alfalfa
function growingradius
function recfastConstructor
function recfastConstructor
module intergalactic_medium_state_internal_evolver
function intergalactic_medium_state_internal_odes
subroutine merger_tree_read_initialize
subroutine galacticusopen
subroutine merger_trees_simple_process
subroutine node_component_hot_halo_standard_rate_compute
subroutine node_component_hot_halo_cold_mode_outflow_return
subroutine node_component_hot_halo_cold_mode_outflow_return
subroutine node_component_hot_halo_standard_outflow_return
subroutine node_component_hot_halo_standard_outflow_return
subroutine node_component_hot_halo_very_simple_tree_initialize
module radiation_intergalactic_background_internal
subroutine critical_overdensity_kitayama_suto1996_initialize
function halo_mass_fraction_integrated
function importerunitconvert1d
function importerunitconvertscalar
subroutine sussingopen
subroutine merger_trees_simple_process
subroutine node_component_black_hole_standard_rate_compute
subroutine node_component_hot_halo_cold_mode_outflow_return
subroutine node_component_hot_halo_cold_mode_outflow_return
subroutine node_component_hot_halo_standard_outflow_return
subroutine node_component_hot_halo_standard_outflow_return
subroutine node_component_hot_halo_very_simple_tree_initialize
module radiation_intergalactic_background_internal
subroutine critical_overdensity_kitayama_suto1996_initialize
function halo_mass_fraction_integrated
function halo_mass_function_sheth_tormen_differential
module linear_growth_simple
function power_spectrum
19.1. Program units

function power_spectrum_nonlinear_cosmicemu
function power_spectrum_window_function_top_hat
subroutine power_spectrum_window_functions_sharp_kspace_initialize
function power_spectrum_window_function_th_kss_hybrid
subroutine power_spectrum_window_functions_th_kss_hybridInitialize
subroutine transfer_function_bbks_make
subroutine transfer_function_eisenstein_hu_make
function transfer_function_half_mode_mass_eisenstein_hu
function bryannorman1998defaultconstructor
function bryannorman1998defaultconstructor
program test_nfw96_concentration_dark_energy
program test_prada2011_concentration
program tests_comoving_distance
tinker
program tests_halo_mass_function_tinker
program tests_power_spectrum
program tests_sigma

interface: cosmologyparameters

Code lines: 3
Contained by: module cosmology_parameters

type: cosmologyparametersclass

Code lines: 70
Contained by: module cosmology_parameters

function: cosmologyparametersconstructordefault

Description: Return a pointer to the default cosmologyParameters object.
Code lines: 8
Contained by: module cosmology_parameters

function: cosmologyparametersconstructornamed

Description: Return a pointer to a newly created cosmologyParameters object of the specified type.
Code lines: 23
Contained by: module cosmology_parameters
Modules used: galacticus_error iso_varying_string

function: cosmologyparametersdensitycriticalnull

Description: Return the critical density at the present day in units of M⊙/Mpc³.
Code lines: 7
Contained by: module cosmology_parameters
Modules used: galacticus_error

function: cosmologyparametershubbleconstantnull

Description: Return the Hubble constant at the present day. The optional units argument specifies if the return value should be in units of km/s/Mpc (unitsStandard), Gyr⁻¹ (unitsTime), or 100 km/s/Mpc (unitsLittleH).
Code lines: 8
Contained by: module cosmology_parameters
19. Source Code Documentation

Modules used: galacticus_error

subroutine: cosmologyparametersinitialize
    Description: Initialize the default cosmologyParameters object.
    Code lines: 43
    Contained by: module cosmology_parameters
    Modules used: galacticus_error input_parameters iso_varying_string

function: cosmologyparametersisfinalizable
    Description: Return true if this object can be finalized.
    Code lines: 7
    Contained by: module cosmology_parameters

function: cosmologyparametersomegabaryonnull
    Description: Return the cosmological baryon density in units of the critical density at the present day.
    Code lines: 7
    Contained by: module cosmology_parameters
    Modules used: galacticus_error

function: cosmologyparametersomegacurvaturenull
    Description: Return the cosmological curvature density in units of the critical density at the present day.
    Code lines: 7
    Contained by: module cosmology_parameters
    Modules used: galacticus_error

function: cosmologyparametersomegdarkenergynull
    Description: Return the cosmological dark energy density in units of the critical density at the present day.
    Code lines: 7
    Contained by: module cosmology_parameters
    Modules used: galacticus_error

function: cosmologyparametersomegamatternull
    Description: Return the cosmological matter density in units of the critical density at the present day.
    Code lines: 7
    Contained by: module cosmology_parameters
    Modules used: galacticus_error

function: cosmologyparametersomegaradiationnull
    Description: Return the cosmological radiation density in units of the critical density at the present day.
    Code lines: 7
    Contained by: module cosmology_parameters
    Modules used: galacticus_error

interface: cosmologyparameterssimple
    Description: Constructors for the simple cosmological parameters class.
    Code lines: 4
    Contained by: module cosmology_parameters
function: cosmologyparameterssimpleconstructor
*Description:* User-defined constructor for the simple cosmological parameters class.
*Code lines:* 12
*Contained by:* module cosmology_parameters

function: cosmologyparameterssimpledefaultconstructor
*Description:* Default constructor for the simple cosmological parameters class.
*Code lines:* 70
*Contained by:* module cosmology_parameters

subroutine: cosmologyparameterssimpledestructor
*Description:* Default constructor for the simple cosmological parameters class.
*Code lines:* 7
*Contained by:* module cosmology_parameters

function: cosmologyparameterstemperaturecmbnull
*Description:* Return the temperature of the cosmic microwave background radiation (in units of Kelvin) at the present day.
*Code lines:* 7
*Contained by:* module cosmology_parameters

function: densitycriticalsimple
*Description:* Return the present day critical density of the Universe in units of $M_\odot/Mpc^3$.
*Code lines:* 9
*Contained by:* module cosmology_parameters

function: hubbleconstantsimple
*Description:* Return the present day value of the Hubble constant.
*Code lines:* 21
*Contained by:* module cosmology_parameters

function: omegabaryonsimple
*Description:* Return the cosmological baryon density in units of the critical density at the present day.
*Code lines:* 8
*Contained by:* module cosmology_parameters

function: omegacurvaturessimple
*Description:* Return the cosmological curvature density in units of the critical density at the present day.
*Code lines:* 7
*Contained by:* module cosmology_parameters

function: omegadarkenergysimple
*Description:* Return the cosmological dark energy density in units of the critical density at the present day.
19. Source Code Documentation

Code lines: 8
Contained by: module cosmology_parameters
Modules used: galacticus_error

function: omegamattersimple
Description: Return the cosmological matter density in units of the critical density at the present day.
Code lines: 8
Contained by: module cosmology_parameters
Modules used: galacticus_error

function: omegaradiationsimple
Description: Return the cosmological radiation density in units of the critical density at the present day.
Code lines: 9
Contained by: module cosmology_parameters
Modules used: numerical_constants_astronomical numerical_constants_physical

function: temperaturecmbsimple
Description: Return the present day temperature of the CMB.
Code lines: 7
Contained by: module cosmology_parameters

file: cosmology.parameters.simple.F90
Description: A simple implementation of the cosmological parameters class.
Code lines: 233

interface: cosmologyparameterssimple
Description: Constructors for the simple cosmological parameters class.
Code lines: 4
Contained by: file cosmology.parameters.simple.F90

function: cosmologyparameterssimpleconstructor
Description: User-defined constructor for the simple cosmological parameters class.
Code lines: 12
Contained by: file cosmology.parameters.simple.F90

function: cosmologyparameterssimpledefaultconstructor
Description: Default constructor for the simple cosmological parameters class.
Code lines: 70
Contained by: file cosmology.parameters.simple.F90
Modules used: galacticus_display input_parameters

subroutine: cosmologyparameterssimpleDestructor
Description: Default constructor for the simple cosmological parameters class.
Code lines: 7
Contained by: file cosmology.parameters.simple.F90

function: densitycriticalsimple
Description: Return the present day critical density of the Universe in units of $M_\odot$/Mpc$^3$.
Code lines: 9
19.1. Program units

```
Contained by: file cosmology.parameters.simple.F90
Modules used: numerical_constants_math numerical_constants_physical

function: hubbleconstantsimple
Description: Return the present day value of the Hubble constant.
Code lines: 21
Contained by: file cosmology.parameters.simple.F90
Modules used: numerical_constants_math numerical_constants_physical

function: omegabaryonsimple
Description: Return the cosmological baryon density in units of the critical density at the present day.
Code lines: 8
Contained by: file cosmology.parameters.simple.F90
Modules used: numerical_constants_astronomical numerical_constants_prefixes

function: omegacurvature_simple
Description: Return the cosmological curvature density in units of the critical density at the present day.
Code lines: 7
Contained by: file cosmology.parameters.simple.F90

function: omegadarkenergysimple
Description: Return the cosmological dark energy density in units of the critical density at the present day.
Code lines: 8
Contained by: file cosmology.parameters.simple.F90
Modules used: galacticus_error

function: omegamattersimple
Description: Return the cosmological matter density in units of the critical density at the present day.
Code lines: 8
Contained by: file cosmology.parameters.simple.F90
Modules used: galacticus_error

function: omegaradiationsimple
Description: Return the cosmological radiation density in units of the critical density at the present day.
Code lines: 9
Contained by: file cosmology.parameters.simple.F90
Modules used: numerical_constants_astronomical numerical_constants_physical

function: temperaturecmbsimple
Description: Return the present day temperature of the CMB.
Code lines: 7
Contained by: file cosmology.parameters.simple.F90

file: dark_matter_halos.formation_times.F90
Description: Contains a module which implements calculations of dark matter halo formation times.
Code lines: 64

module: dark_matter_halo_formation_times
```
19. Source Code Documentation

**Description:** Implements calculations of dark matter halo formation times.

**Code lines:** 44

**Contained by:** file `dark_matter_halos.formation_times.F90`

**Used by:**
- function `zhao2009concentration`
- subroutine `node_component_mass_flow_statistics_standard_merger_tree_init`
- subroutine `node_component_merging_statistics_standard_merger_tree_init_set`

**function: dark_matter_halo_formation_time**

**Description:** Returns the time at which the main branch progenitor of `thisNode` first had a mass equal to `formationMassFraction` of the current mass.

**Code lines:** 34

**Contained by:** module `dark_matter_halo_formation_times`

**Modules used:**
- `dark_matter_halo_mass_accretion_histories`

**file: dark_matter_halos.mass_accretion_history.F90**

**Description:** Contains a module which implements calculations of dark matter halo mass accretion histories.

**Code lines:** 117

**module: dark_matter_halo_mass_accretion_histories**

**Description:** Implements calculations of dark matter halo mass accretion histories.

**Code lines:** 97

**Contained by:** file `dark_matter_halos.mass_accretion_history.F90`

**Modules used:**
- `galacticus_nodes`
- `iso_varying_string`

**Used by:**
- function `dark_matter_halo_formation_time`
- subroutine `merger_tree_smooth_accretion_do`
- program `test_zhao2009_flat`
- program `test_zhao2009_dark_energy`
- program `test_zhao2009_open`

**function: dark_matter_halo_mass_accretion_time**

**Description:** Returns the time for `thisNode` in `thisTree` according to the mass accretion history.

**Code lines:** 13

**Contained by:** module `dark_matter_halo_mass_accretion_histories`

**subroutine: dark_matter_mass_accretion_initialize**

**Description:** Initialize the dark matter mass accretion history module.

**Code lines:** 54

**Contained by:** module `dark_matter_halo_mass_accretion_histories`

**Modules used:**
- `dark_matter_halo_mass_accretion_histories_wechsler2002`
- `dark_matter_halo_mass_accretion_histories_zhao2009`
- `galacticus_error`
- `input_parameters`


**Description:** Contains a module which implements the Wechsler et al. [2002] halo mass accretion algorithm.

**Code lines:** 143

**module: dark_matter_halo_mass_accretion_histories_wechsler2002**
19.1. Program units

Description: Implements the Wechsler et al. [2002] halo mass accretion algorithm.
Code lines: 123


Modules used: galacticus_nodes

Used by: subroutine dark_matter_mass_accretion_initialize

function: dark_matter_halo_mass_accretion_time_wechsler2002
Description: Compute the time corresponding to nodeMass in the mass accretion history of thisNode using the algorithm of Wechsler et al. [2002].
Code lines: 33

Contained by: module dark_matter_halo_mass_accretion_histories_wechsler2002

Modules used: cosmology_functions

subroutine: dark_matter_mass_accretion_wechsler2002_initialize
Description: Initializes the “Wechsler2002” mass accretion history module.
Code lines: 41

Contained by: module dark_matter_halo_mass_accretion_histories_wechsler2002

Modules used: input_parameters iso_varying_string

function: expansion_factor_at_formation
Description: Computes the expansion factor at formation using the simple model of Bullock et al. [2001].
Code lines: 27

Contained by: module dark_matter_halo_mass_accretion_histories_wechsler2002

Modules used: cosmology_functions critical_overdensity power_spectra

file: dark_matter_halos.mass_accretion_history.Zhao2009.F90
Description: Contains a module which implements the Zhao et al. [2009] halo mass accretion algorithm.
Code lines: 158

module: dark_matter_halo_mass_accretion_histories_zhao2009
Description: Implements the Zhao et al. [2009] halo mass accretion algorithm.
Code lines: 138

Contained by: file dark_matter_halos.mass_accretion_history.Zhao2009.F90

Modules used: fgsl galacticus_nodes iso_c_binding

Used by: subroutine dark_matter_mass_accretion_initialize

function: dark_matter_halo_mass_accretion_time_zhao2009
Description: Compute the time corresponding to nodeMass in the mass accretion history of thisNode using the algorithm of Zhao et al. [2009].
Code lines: 51

Contained by: module dark_matter_halo_mass_accretion_histories_zhao2009

Modules used: critical_overdensity galacticus_error ode_solver power_spectra

subroutine: dark_matter_mass_accretion_zhao2009_initialize
**Description:** Initializes the “Zhao2009” mass accretion history module.

**Code lines:** 13

**Contained by:** module `dark_matter_halo_mass_accretion_histories_zhao2009`

**Modules used:** `iso_varying_string`

**function:** `growthrateodes`

**Description:** System of differential equations to solve for the growth rate.

**Code lines:** 43

**Contained by:** module `dark_matter_halo_mass_accretion_histories_zhao2009`

**Modules used:** `critical_overdensity` `power_spectra`

**file:** `dark_matter_halos.mass_loss_rates.F90`

**Description:** Contains a module which implements calculations of mass loss rates from dark matter halos.

**Code lines:** 115

**module:** `dark_matter_halos_mass_loss_rates`

**Description:** Implements calculations of mass loss rates from dark matter halos.

**Code lines:** 95

**Contained by:** file `dark_matter_halos.mass_loss_rates.F90`

**Modules used:** `galacticus_nodes` `iso_varying_string`

**Used by:** subroutine `node_component_satellite_standard_rate_compute`

**subroutine:** `dark_matter_halo_mass_loss_rates_initialize`

**Description:** Initialize the dark matter halos mass loss rate module.

**Code lines:** 54

**Contained by:** module `dark_matter_halos_mass_loss_rates`

**Modules used:** `dark_matter_halos_mass_loss_rates_null` `dark_matter_halos_mass_loss_rates_vandenbosch` `galacticus_error` `input_parameters`

**function:** `dark_matter_halos_mass_loss_rate`

**Description:** Returns the rate of mass loss (in $M_\odot$/Gyr) from thisNode.

**Code lines:** 12

**Contained by:** module `dark_matter_halos_mass_loss_rates`

**file:** `dark_matter_halos.mass_loss_rates.null.F90`

**Description:** Contains a module which implements a null calculation of dark matter halo mass loss rates.

**Code lines:** 52

**module:** `dark_matter_halos_mass_loss_rates_null`

**Description:** Implements a null calculation of dark matter halo mass loss rates.

**Code lines:** 32

**Contained by:** file `dark_matter_halos.mass_loss_rates.null.F90`

**Used by:** subroutine `dark_matter_halo_mass_loss_rates_initialize`

**function:** `dark_matter_halos_mass_loss_rate_null`

**Description:** Returns the a zero rate of mass loss from dark matter halos.
19.1. Program units

Code lines: 8
Contained by: module dark_matter_halos_mass_loss_rates_null
Modules used: galacticus_nodes

subroutine: dark_matter_halos_mass_loss_rate_null_initialize
Description: Initializes the “null” dark matter halo mass loss rate method.
Code lines: 9
Contained by: module dark_matter_halos_mass_loss_rates_null
Modules used: iso_varying_string

file: dark_matter_halos.mass_loss_rates.vanDenBosch.F90
Description: Contains a module which implements a calculation of dark matter halo mass loss rates using the method of van den Bosch et al. [2005].
Code lines: 93

module: dark_matter_halos_mass_loss_rates_vandenbosch
Description: Implements a calculation of dark matter halo mass loss rates using the method of van den Bosch et al. [2005].
Code lines: 72
Contained by: file dark_matter_halos.mass_loss_rates.vanDenBosch.F90
Used by: subroutine dark_matter_halo_mass_loss_rates_initialize

function: dark_matter_halos_mass_loss_rate_vandenbosch
Description: Returns the rate of mass loss from dark matter halos using the prescription of van den Bosch et al. [2005].
Code lines: 29
Contained by: module dark_matter_halos_mass_loss_rates_vandenbosch
Modules used: cosmology_functions galacticus_nodes virial_density_contrast

subroutine: dark_matter_halos_mass_loss_rate_vandenbosch_initialize
Description: Initializes the “vanDenBosch2005” dark matter halo mass loss rate method.
Code lines: 21
Contained by: module dark_matter_halos_mass_loss_rates_vandenbosch
Modules used: cosmology_functions iso_varying_string virial_density_contrast

file: dark_matter_halos.scales.F90
Description: Contains a module which provides a class implementing scales of dark matter halos.
Code lines: 791

module: dark_matter_halo_scales
Description: Provides a class implementing scales of dark matter halos. An implementation of dark matter halo scales based on virial density contrast.
Code lines: 771
Contained by: file dark_matter_halos.scales.F90
Modules used: fgsl galacticus_nodes iso_varying_string kind_numbers tables virial_density_contrast
19. Source Code Documentation

**Used by:**
- function `coldmodechemicalmasses`
- function `coldmodecoldmodefraction`
- function `coldmodefailedaccretedmass`
- function `coldmodefailedaccretionrate`
- function `coldmodechemicalmasses`
- function `coldmodecoldmodefraction`
- function `coldmodefailedaccretedmass`
- function `coldmodefailedaccretionrate`
- function `black_hole_binary_initial_radius_volonteri_2003`
- function `cooling_cold_mode_infall_rate_dynamical_time`
- function `cooling_radius_isothermal`
- subroutine `cooling_rate_modifier_cut_off`
- function `infall_radius_growth_rate_cooling_freefall`
- function `cooling_time_available_wf`
- file `dark_matter_profiles.Einasto.F90`
- function `einastodensity`
- function `einastoenergy`
- function `einastoenergygrowthrate`
- function `einastokspace`
- function `einastopotential`
- function `einastorotationnormalization`
- module `dark_matter_profiles`
- function `einastodensity`
- function `einastoenergy`
- function `einastoenergygrowthrate`
- function `einastokspace`
- function `einastopotential`
- function `einastorotationnormalization`
- function `isothermalcircularvelocity`
- function `isothermalcircularvelocitymaximum`
- function `isothermalenclosedmass`
- function `isothermalenergy`
- function `isothermalenergygrowthrate`
- function `isothermalfreefallradius`
- function `isothermalfreefallradiusincreaserate`
- function `isothermalkspace`
- function `isothermalrotationnormalization`
- function `isothermalradiusfromspecificangularmomentum`
- function `isothermalrotationnormalization`
- function `nfwdensity`
- function `nfwenclosedmass`
- function `nfwenergy`
- function `nfwenergygrowthrate`
- function `nfwfreefallradius`
- function `nfwfreefallradiusincreaserate`
- function `nfwkspase`
- function `nfwpotential`
- function `nfwrotationnormalization`
- file `dark_matter_profiles.NFW.F90`
- function `nfwdensity`
- function `nfwenclosedmass`
- function `nfwenergy`
- function `nfwenergygrowthrate`
- function `nfwfreefallradius`
- function `nfwfreefallradiusincreaserate`
- function `nfwkspase`
- function `nfwpotential`
- function `nfwrotationnormalization`
- file `dark_matter_profiles.isothermal.F90`
- function `isothermalcircularvelocity`
- function `isothermalcircularvelocitymaximum`
function isothermal_density
function isothermal_energy
function isothermal_freefall_radius
function isothermal_volume
function isothermal_rotation_normalization
function isothermal_radius_from_specific_angular_momentum
function diemer_kravtsov_2014_dark_matter_profile_definition
function diemer_kravtsov_2014_dark_matter_profile_definition
function gao_2008_dark_matter_profile_definition
function nfw_1996_dark_matter_profile_definition
function zhao_2009_dark_matter_profile_definition
function munozcuartas_2011_dark_matter_profile_definition
function prada_2011_dark_matter_profile_definition
function zhao_2009_dark_matter_profile_definition
function nfw_1996_dark_matter_profile_definition
function prada_2011_dark_matter_profile_definition
function zhao_2009_dark_matter_profile_definition
function nfw_1996_dark_matter_profile_definition
subroutine solve_for_radius
subroutine solve_for_radius
subroutine galacticus_output_tree_density_contrast
subroutine galacticus_output_tree_velocity_dispersion
subroutine galacticus_state_store
function hot_halo_cold_mode_density_core_vfrac
function hot_halo_RAM_pressure_stripping_font_2008_get
function hot_halo_temperature_virial_get
function growing_radius
function generating_radius
function hot_halo_RAM_pressure_stripping_font_2008_get
function hot_halo_temperature_virial_get
source code documentation

subroutine assign_scale_radii
subroutine merger_tree_structure_output
subroutine node_component_black_hole_-simple_rate_compute
subroutine node_component_black_hole_-standard_rate_compute
subroutine node_component_dark_matter_-profile_scale_rate_compute
subroutine node_component_dark_matter_-profile_scale_preset_rate_compute
subroutine node_component_disk_-exponential_radius_solver_-plausibility
subroutine node_component_disk_-simple_rate_compute
subroutine node_component_disk_-standard_rate_compute
subroutine node_component_disk_-cold_mode_node_merger
subroutine node_component_disk_-cold_mode_promote
subroutine node_component_disk_-cold_mode_satellite_merger
subroutine node_component_disk_-outflow_tracking_rate_compute
subroutine node_component_disk_-standard_formation
subroutine node_component_disk_-standard_initializer
function node_component_disk_-standard_outer_radius
function node_component_disk_-standard_outflow_stripped_fraction
subroutine node_component_disk_-standard_promote
subroutine node_component_disk_-standard_rate_compute
subroutine node_component_disk_-standard_scale_set
subroutine node_component_merging_-statistics_recent_node_merger
subroutine node_component_satellite_-orbiting_scale_set
function satellite_dynamical_friction_-acceleration_chandrasekhar
subroutine scan_for_mergers
subroutine merger_trees_render_dump
function hot_mode_fraction
subroutine node_component_dark_matter_-profile_scale_initialize_scale
function node_component_dark_matter_-profile_scale_scale
subroutine node_component_disk_-exponential_post_evolve
subroutine node_component_disk_-exponential_rate_compute
function node_component_disk_-simple_sfr
subroutine node_component_dynamics_-statistics_bars_timestep
subroutine node_component_hot_halo_-cold_mode_formation
subroutine node_component_hot_halo_-cold_mode_outflow_return
subroutine node_component_hot_halo_-cold_mode_rate_compute
subroutine node_component_hot_halo_-cold_mode_scale_set
subroutine node_component_hot_halo_-cold_mode_satellite_merger
subroutine node_component_hot_halo_-cold_mode_outflow_tracking_rate_compute
subroutine node_component_hot_halo_-standard_formation
subroutine node_component_hot_halo_-standard_initializer
function node_component_hot_halo_-standard_outer_radius
function node_component_hot_halo_-standard_outflow_stripped_fraction
subroutine node_component_hot_halo_-standard_promote
subroutine node_component_hot_halo_-standard_rate_compute
subroutine node_component_hot_halo_-standard_scale_set
subroutine node_component_merging_-statistics_recent_node_merger
subroutine node_component_satellite_-orbiting_scale_set
function satellite_dynamical_friction_-acceleration_chandrasekhar
subroutine node_component_dark_matter_-profile_scale_initialize_scale
function node_component_dark_matter_-profile_scale_scale
subroutine node_component_disk_-exponential_post_evolve
subroutine node_component_disk_-exponential_rate_compute
function node_component_disk_-simple_sfr
subroutine node_component_dynamics_-statistics_bars_timestep
subroutine node_component_hot_halo_-cold_mode_formation
subroutine node_component_hot_halo_-cold_mode_outflow_return
subroutine node_component_hot_halo_-cold_mode_rate_compute
subroutine node_component_hot_halo_-cold_mode_scale_set
subroutine node_component_hot_halo_-cold_mode_satellite_merger
subroutine node_component_hot_halo_-cold_mode_outflow_tracking_rate_compute
subroutine node_component_hot_halo_-standard_formation
subroutine node_component_hot_halo_-standard_initializer
function node_component_hot_halo_-standard_outer_radius
function node_component_hot_halo_-standard_outflow_stripped_fraction
subroutine node_component_hot_halo_-standard_promote
subroutine node_component_hot_halo_-standard_rate_compute
subroutine node_component_hot_halo_-standard_scale_set
subroutine node_component_merging_-statistics_recent_node_merger
subroutine node_component_satellite_-orbiting_scale_set
function satellite_dynamical_friction_-acceleration_chandrasekhar
function boylankolchin2008timeuntilmerging
function jiang2008timeuntilmerging
function laceycole1993timeuntilmerging
function laceycole1993timeuntilmergingmassdependence
function laceycole1993timeuntilmergingmassdependence
function boylankolchin2008timeuntilmerging
function jiang2008timeuntilmerging
function laceycole1993timeuntilmerging
function laceycole1993timeuntilmergingmassdependence
function laceycole1993timeuntilmergingmassdependence
function virial_orbital_parameters_-benson2005
function virial_orbital_parameters_-fixed
function satellite_tidal_heating_rate_-gnedin
function star_formation_feedback_disk_-outflow_rate_halo_scaling
function star_formation_timescale_-disk_halo_scaling

interface: darkmatterhaloscale
Code lines: 3
Contained by: module dark_matter_halo_scales

subroutine: darkmatterhaloscalecalculationreset
Description: Reset the calculation state of the object.
Code lines: 7
Contained by: module dark_matter_halo_scales

type: darkmatterhaloscaleclass
Code lines: 91
Contained by: module dark_matter_halo_scales

function: darkmatterhaloscaleconstructordefault
Description: Return a pointer to the default darkMatterHaloScale object.
Code lines: 8
Contained by: module dark_matter_halo_scales

function: darkmatterhaloscaleconstructornamed
Description: Return a pointer to a newly created darkMatterHaloScale object of the specified type.
Code lines: 23
Contained by: module dark_matter_halo_scales
Modules used: galacticus_error iso_varying_string

subroutine: darkmatterhaloscalecalculationreset
Description: Store the state to file.
Code lines: 9
Contained by: module dark_matter_halo_scales

subroutine: darkmatterhaloscaledostateretrieve
Description: Retrieve the state from file.
Code lines: 10
Contained by: module dark_matter_halo_scales
subroutine: darkmatterhaloscaledostatestore
Description: Store the state to file.
Code lines: 10
Contained by: module dark_matter_halo_scales

function: darkmatterhaloscaledynamicaltimescalenull
Description: The characteristic dynamical timescale of a dark matter halo.
Code lines: 8
Contained by: module dark_matter_halo_scales
Modules used: galacticus_error

subroutine: darkmatterhaloscaleinitialize
Description: Initialize the default darkMatterHaloScale object.
Code lines: 40
Contained by: module dark_matter_halo_scales
Modules used: galacticus_error, input_parameters, iso_varying_string

function: darkmatterhaloscaleisfinalizable
Description: Return true if this object can be finalized.
Code lines: 7
Contained by: module dark_matter_halo_scales

function: darkmatterhaloscalemeandensitygrowthratenull
Description: The growth rate of the mean density of a dark matter halo.
Code lines: 8
Contained by: module dark_matter_halo_scales
Modules used: galacticus_error

function: darkmatterhaloscalemeandensitynull
Description: The mean density of a dark matter halo.
Code lines: 8
Contained by: module dark_matter_halo_scales
Modules used: galacticus_error

subroutine: darkmatterhaloscalestaterestore
Description: Restore the state of the object to file.
Code lines: 9
Contained by: module dark_matter_halo_scales
Modules used: fgsl

subroutine: darkmatterhaloscalestatestore
Description: Store the state of the object to file.
Code lines: 9
Contained by: module dark_matter_halo_scales
Modules used: fgsl

interface: darkmatterhaloscalevirialdensitycontrastdefinition
19.1. Program units

Description: Constructors for the \texttt{virialDensityContrastDefinition} dark matter halo scales class.
Code lines: 4
Contained by: module \texttt{dark_matter_halo_scales}

\textbf{function: darkmatterhaloscalevirialradiusgrowthratenull}
Description: The growth rate of the virial radius of a dark matter halo.
Code lines: 8
Contained by: module \texttt{dark_matter_halo_scales}
Modules used: \texttt{galacticus_error}

\textbf{function: darkmatterhaloscalevirialradiusnull}
Description: The virial radius of a dark matter halo.
Code lines: 8
Contained by: module \texttt{dark_matter_halo_scales}
Modules used: \texttt{galacticus_error}

\textbf{function: darkmatterhaloscalevirialtemperaturenull}
Description: The virial temperature of a dark matter halo.
Code lines: 8
Contained by: module \texttt{dark_matter_halo_scales}
Modules used: \texttt{galacticus_error}

\textbf{function: darkmatterhaloscalevirialvelocitygrowthratenull}
Description: The growth rate of the virial velocity of a dark matter halo.
Code lines: 8
Contained by: module \texttt{dark_matter_halo_scales}
Modules used: \texttt{galacticus_error}

\textbf{function: darkmatterhaloscalevirialvelocitynull}
Description: The virial velocity of a dark matter halo.
Code lines: 8
Contained by: module \texttt{dark_matter_halo_scales}
Modules used: \texttt{galacticus_error}

\textbf{subroutine: virialdensitycontrastdefinitioncalculationreset}
Description: Reset the halo scales calculation.
Code lines: 12
Contained by: module \texttt{dark_matter_halo_scales}

\textbf{function: virialdensitycontrastdefinitionconstructor}
Description: Default constructor for the \texttt{virialDensityContrastDefinition} dark matter halo scales class.
Code lines: 17
Contained by: module \texttt{dark_matter_halo_scales}

\textbf{function: virialdensitycontrastdefinitiondefaultconstructor}
Description: Default constructor for the \texttt{virialDensityContrastDefinition} dark matter halo scales class.
Code lines: 9
Contained by: module \texttt{dark_matter_halo_scales}
subroutine: virialdensitycontrastdefinitiondestructor
Description: Destructor for the virialDensityContrastDefinition dark matter halo scales class.
Code lines: 8
Contained by: module dark_matter_halo_scales

function: virialdensitycontrastdefinitiondynamicaltimescale
Description: Returns the dynamical timescale for thisNode.
Code lines: 17
Contained by: module dark_matter_halo_scales
Modules used: numerical_constants_astronomical

function: virialdensitycontrastdefinitionmeandensity
Description: Returns the mean density for thisNode.
Code lines: 41
Contained by: module dark_matter_halo_scales
Modules used: cosmology_functions, cosmology_parameters

function: virialdensitycontrastdefinitionmeandensitygrowthrate
Description: Returns the growth rate of the mean density for thisNode.
Code lines: 33
Contained by: module dark_matter_halo_scales
Modules used: cosmology_functions

subroutine: virialdensitycontrastdefinitionstaterestore
Description: Retrieve the tabulation state from the file.
Code lines: 12
Contained by: module dark_matter_halo_scales
Modules used: fgsl

subroutine: virialdensitycontrastdefinitionstatestore
Description: Write the tabulation state to file.
Code lines: 10
Contained by: module dark_matter_halo_scales
Modules used: fgsl

function: virialdensitycontrastdefinitionvirialradius
Description: Returns the virial radius scale for thisNode.
Code lines: 22
Contained by: module dark_matter_halo_scales
Modules used: numerical_constants_math

function: virialdensitycontrastdefinitionvirialradiusgrowthrate
Description: Returns the growth rate of the virial radius scale for thisNode.
Code lines: 11
Contained by: module dark_matter_halo_scales

function: virialdensitycontrastdefinitionvirialtemperature
Description: Returns the virial temperature (in Kelvin) for thisNode.
Code lines: 18
Contained by: module dark_matter_halo_scales
Modules used: numerical_constants_astronomical numerical_constants_physical

function: virialdensitycontrastdefinitionvirialvelocity
Description: Returns the virial velocity scale for thisNode.
Code lines: 22
Contained by: module dark_matter_halo_scales
Modules used: numerical_constants_physical

function: virialdensitycontrastdefinitionvirialvelocitygrowthrate
Description: Returns the growth rate of the virial velocity scale for thisNode.
Code lines: 11
Contained by: module dark_matter_halo_scales

file: dark_matter_halos.scales.virial_density_contrast.F90
Description: An implementation of dark matter halo scales based on virial density contrast.
Code lines: 336
Modules used: kind_numbers tables virial_density_contrast

interface: darkmatterhaloscalevirialdensitycontrastdefinition
Description: Constructors for the virialDensityContrastDefinition dark matter halo scales class.
Code lines: 4
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90

subroutine: virialdensitycontrastdefinitioncalculationreset
Description: Reset the halo scales calculation.
Code lines: 12
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90

function: virialdensitycontrastdefinitionconstructor
Description: Default constructor for the virialDensityContrastDefinition dark matter halo scales class.
Code lines: 17
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90

function: virialdensitycontrastdefinitiondefaultconstructor
Description: Default constructor for the virialDensityContrastDefinition dark matter halo scales class.
Code lines: 9
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90

subroutine: virialdensitycontrastdefinitiondestructor
Description: Destructor for the virialDensityContrastDefinition dark matter halo scales class.
Code lines: 8
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90
function: virialdensitycontrastdefinitiondynamicaltimescale
Description: Returns the dynamical timescale for thisNode.
Code lines: 17
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90
Modules used: numerical_constants_astronomical

function: virialdensitycontrastdefinitionmeandensity
Description: Returns the mean density for thisNode.
Code lines: 41
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90
Modules used: cosmology_functions cosmology_parameters

function: virialdensitycontrastdefinitionmeandensitygrowthrate
Description: Returns the growth rate of the mean density for thisNode.
Code lines: 33
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90
Modules used: cosmology_functions

subroutine: virialdensitycontrastdefinitionstaterestore
Description: Retrieve the tabulation state from the file.
Code lines: 12
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90
Modules used: fgsl

subroutine: virialdensitycontrastdefinitionstatestore
Description: Write the tabulation state to file.
Code lines: 10
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90
Modules used: fgsl

function: virialdensitycontrastdefinitionvirialradius
Description: Returns the virial radius scale for thisNode.
Code lines: 22
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90
Modules used: numerical_constants_math

function: virialdensitycontrastdefinitionvirialradiusgrowthrate
Description: Returns the growth rate of the virial radius scale for thisNode.
Code lines: 11
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90

function: virialdensitycontrastdefinitionvirialtemperature
Description: Returns the virial temperature (in Kelvin) for thisNode.
Code lines: 18
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90
Modules used: numerical_constants_astronomical numerical_constants_physical
function: virialdensitycontrastdefinitionvirialvelocity
Description: Returns the virial velocity scale for thisNode.
Code lines: 22
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90
Modules used: numerical_constants_physical

function: virialdensitycontrastdefinitionvirialvelocitygrowthrate
Description: Returns the growth rate of the virial velocity scale for thisNode.
Code lines: 11
Contained by: file dark_matter_halos.scales.virial_density_contrast.F90

file: dark_matter_halos.spins.F90
Description: Contains a module which implements calculations of dark matter halo angular momentum.
Code lines: 94

module: dark_matter_halo_spins
Description: Implements calculations of dark matter halo angular momentum.
Code lines: 74
Contained by: file dark_matter_halos.spins.F90
Used by: subroutine node_component_hot_halo_-cold_mode_rate_compute subroutine node_component_hot_halo_-cold_mode_tree_initialize subroutine node_component_hot_halo_-standard_rate_compute subroutine node_component_hot_halo_-standard_tree_initialize

function: dark_matter_halo_angular_momentum
Description: Returns the total angular momentum of thisNode based on its mass, energy and spin parameter.
Code lines: 19
Contained by: module dark_matter_halo_spins
Modules used: dark_matter_profiles galacticus_nodes numerical_constants_physical

function: dark_matter_halo_angular_momentum_growth_rate
Description: Returns the rate of change of the total angular momentum of thisNode based on its mass, energy and spin parameter.
Code lines: 19
Contained by: module dark_matter_halo_spins
Modules used: dark_matter_profiles galacticus_nodes

subroutine: dark_matter_halo_spins_initialize
Description: Initialize the halo spins module.
Code lines: 19
Contained by: module dark_matter_halo_spins
Modules used: galacticus_error galacticus_nodes iso_varying_string

Description: Contains a module which implements the Bett et al. [2007] halo spin distribution.
module: halo_spin_distributions_bett2007
Description: Implements the Bett et al. [2007] halo spin distribution.
Code lines: 137
Contained by: module halo_spin_distributions
Modules used: fgsl, tables
Used by: function halo_spin_distribution_sample subroutine galacticus_state_retrieve subroutine galacticus_state_store

function: halo_spin_distribution_bett2007
Description: Return a halo spin from a lognormal distribution.
Code lines: 13
Contained by: module halo_spin_distributions_bett2007
Modules used: galacticus_nodes, pseudo_random

subroutine: halo_spin_distribution_bett2007_initialize
Description: Initializes the “Bett2007” halo spin distribution module.
Code lines: 49
Contained by: module halo_spin_distributions_bett2007
Modules used: gamma_functions, input_parameters, iso_varying_string

subroutine: halo_spin_distribution_bett2007_snapshot
Description: Store a snapshot of the random number generator internal state.
Code lines: 7
Contained by: module halo_spin_distributions_bett2007

subroutine: halo_spin_distribution_bett2007_state_retrieve
Description: Write the stored snapshot of the random number state to file.
Code lines: 10
Contained by: module halo_spin_distributions_bett2007
Modules used: pseudo_random

subroutine: halo_spin_distribution_bett2007_state_store
Description: Write the stored snapshot of the random number state to file.
Code lines: 10
Contained by: module halo_spin_distributions_bett2007
Modules used: pseudo_random

Description: Contains a module that implements calculations of dark matter halo spin distributions
Code lines: 107

module: halo_spin_distributions
Description: Implements calculations of dark matter halo spin distributions
Code lines: 87
Contained by: file dark_matter_halos.spins.distributions.F90
Modules used: galacticus_nodes, iso_varying_string
19.1. Program units

**Used by:** subroutine assign_spin_parameters subroutine node_componentSpin_randomize_initialize_spins

**function:** halo_spin_distribution_sample

*Description:* Return a halo spin selected randomly from a distribution.

*Code lines:* 60

*Contained by:* module halo_spin_distributions

*Modules used:*
- galacticus_error
- halo_spin_distributions_bett2007
- halo_spin_distributions_delta_function
- halo_spin_distributions_lognormal
- input_parameters

**interface:** halo_spin_sample_get_template

*Code lines:* 5

*Contained by:* module halo_spin_distributions

**function:** halo_spin_sample_get_template

*Code lines:* 3

*Contained by:* interface halo_spin_sample_get_template

**file:** dark_matter_halos.spins.distributions.delta_function.F90

*Description:* Contains a module which implements a delta function halo spin distribution.

*Code lines:* 69

**module:** halo_spin_distributions_delta_function

*Description:* Implements a delta function halo spin distribution (i.e. all halos have the same spin).

*Code lines:* 49

*Contained by:* file dark_matter_halos.spins.distributions.delta_function.F90

*Used by:* function halo_spin_distribution_sample

**function:** halo_spin_distribution_delta_function

*Description:* Return a halo spin from a delta function distribution.

*Code lines:* 8

*Contained by:* module halo_spin_distributions_delta_function

*Modules used:* galacticus_nodes

**subroutine:** halo_spin_distribution_delta_function_initialize

*Description:* Initializes the “delta function” halo spin distribution module.

*Code lines:* 23

*Contained by:* module halo_spin_distributions_delta_function

*Modules used:* input_parameters iso_varying_string

**file:** dark_matter_halos.spins.distributions.lognormal.F90

*Description:* Contains a module which implements a lognormal halo spin distribution.

*Code lines:* 132

**module:** halo_spin_distributions_lognormal

*Description:* Implements a lognormal halo spin distribution.

*Code lines:* 112
19. Source Code Documentation

**Contained by:** file `dark_matter_halos.spins.distributions.lognormal.F90`
**Modules used:** `fgsl`
**Used by:** function `halo_spin_distribution_sample` subroutine `galacticus_state_retrieve` subroutine `galacticus_state_snapshot` subroutine `galacticus_state_store`

**function:** `halo_spin_distribution_lognormal`
**Description:** Return a halo spin from a lognormal distribution.
**Code lines:** 12
**Contained by:** module `halo_spin_distributions_lognormal`
**Modules used:** `galacticus_nodes` `gaussian_random`

**subroutine:** `halo_spin_distribution_lognormal_initialize`
**Description:** Initializes the “Lognormal” halo spin distribution module.
**Code lines:** 35
**Contained by:** module `halo_spin_distributions_lognormal`
**Modules used:** `input_parameters` `iso_varying_string`

**subroutine:** `halo_spin_distribution_lognormal_snapshot`
**Description:** Store a snapshot of the random number generator internal state.
**Code lines:** 7
**Contained by:** module `halo_spin_distributions_lognormal`

**subroutine:** `halo_spin_distribution_lognormal_state_retrieve`
**Description:** Write the stored snapshot of the random number state to file.
**Code lines:** 10
**Contained by:** module `halo_spin_distributions_lognormal`
**Modules used:** `pseudo_random`

**subroutine:** `halo_spin_distribution_lognormal_state_store`
**Description:** Write the stored snapshot of the random number state to file.
**Code lines:** 10
**Contained by:** module `halo_spin_distributions_lognormal`
**Modules used:** `pseudo_random`

**file:** `dark_matter_profiles.Einasto.F90`
**Description:** An implementation of “Einasto” dark matter halo profiles.
**Code lines:** 1261
**Modules used:** `dark_matter_halo_scales` `kind_numbers` `tables`

**interface:** `darkmatterprofileeinasto`
**Description:** Constructors for the `einasto` dark matter halo profile class.
**Code lines:** 4
**Contained by:** file `dark_matter_profiles.Einasto.F90`

**subroutine:** `einstastocalculationreset`
**Description:** Reset the dark matter profile calculation.
**Code lines:** 8
**Contained by:** file `dark_matter_profiles.Einasto.F90`
function: einastocircularvelocity
Description: Returns the circular velocity (in km/s) in the dark matter profile of node at the given radius (given in units of Mpc).
Code lines: 15
Contained by: file dark_matter_profiles.Einasto.F90
Modules used: numerical_constants_physical

function: einastocircularvelocitymaximum
Description: Returns the maximum circular velocity (in km/s) in the dark matter profile of node.
Code lines: 37
Contained by: file dark_matter_profiles.Einasto.F90
Modules used: numerical_constants_physical root_finder

function: einastocircularvelocitypeakradius
Description: Computes the derivative of the square of circular velocity for an Einasto density profile.
Code lines: 8
Contained by: function einastocircularvelocitymaximum
Modules used: gamma_functions

function: einastoconstructor
Description: Generic constructor for the einasto dark matter halo profile class.
Code lines: 46
Contained by: file dark_matter_profiles.Einasto.F90
Modules used: array_utilities galacticus_error

function: einastodefaultconstructor
Description: Default constructor for the einasto dark matter halo profile class.
Code lines: 8
Contained by: file dark_matter_profiles.Einasto.F90
Modules used: input_parameters

function: einastodensity
Description: Returns the density (in \(M_\odot\) Mpc\(^{-3}\)) in the dark matter profile of node at the given radius (given in units of Mpc).
Code lines: 21
Contained by: file dark_matter_profiles.Einasto.F90
Modules used: dark_matter_halo_scales

function: einastodensityscalefree
Description: Returns the density (in units such that the virial mass and scale length are unity) in an Einasto dark matter profile with given concentration and alpha at the given radius (given in units of the scale radius).
Code lines: 13
Contained by: file dark_matter_profiles.Einasto.F90
Modules used: gamma_functions numerical_constants_math

subroutine: einastodedestructor
Description: Destructor for the einasto dark matter halo profile class.
19. Source Code Documentation

Code lines: 17  
Contained by: file dark_matter_profiles.Einasto.F90  
Modules used: numerical_interpolation

**function**: einastoenclosedmass  
**Description**: Returns the enclosed mass (in $M_\odot$) in the dark matter profile of node at the given radius (given in units of Mpc).

Code lines: 21  
Contained by: file dark_matter_profiles.Einasto.F90  
Modules used: dark_matter_halo_scales

**function**: einastoenclosedmassscalefree  
**Description**: Returns the enclosed mass (in units of the virial mass) in an Einasto dark matter profile with given concentration at the given radius (given in units of the scale radius).

Code lines: 14  
Contained by: file dark_matter_profiles.Einasto.F90  
Modules used: gamma_functions

**function**: einastoenergy  
**Description**: Return the energy of an Einasto halo density profile.

Code lines: 41  
Contained by: file dark_matter_profiles.Einasto.F90  
Modules used: dark_matter_halo_scales iso_c_binding numerical_interpolation

**function**: einastoenergygrowthrate  
**Description**: Return the energy of an Einasto halo density profile.

Code lines: 44  
Contained by: file dark_matter_profiles.Einasto.F90  
Modules used: dark_matter_halo_scales iso_c_binding numerical_interpolation

**subroutine**: einastoenergytablemake  
**Description**: Create a tabulation of the energy of Einasto profiles as a function of their concentration of $\alpha$ parameter.

Code lines: 134  
Contained by: file dark_matter_profiles.Einasto.F90  
Modules used: iso_c_binding memory_management numerical_constants_math numerical_integration numerical_interpolation numerical_ranges

**function**: einastojeansequationintegrand  
**Description**: Integrand for Einasto profile Jeans equation.

Code lines: 10  
Contained by: subroutine einastoenergytablemake  
Modules used: iso_c_binding

**function**: einastokineticenergyintegrand  
**Description**: Integrand for Einasto profile kinetic energy.
19.1. Program units

Code lines: 10
Contained by: subroutine einastoenergytablemake
Modules used: iso_c_binding

**function: einastopotentialenergyintegrand**
Description: Integrand for Einasto profile potential energy.
Code lines: 10
Contained by: subroutine einastoenergytablemake
Modules used: iso_c_binding

**subroutine: einastofourierprofiletablemake**
Description: Create a tabulation of the Fourier transform of Einasto profiles as a function of their $\alpha$ parameter and dimensionless wavenumber.
Code lines: 117
Contained by: file dark_matter_profiles.Einasto.F90
Modules used: galacticus_display iso_c_binding memory_management numerical_integration numerical_interpolation numerical_ranges

**function: einastofourierprofileintegrand**
Description: Integrand for Einasto Fourier profile.
Code lines: 11
Contained by: subroutine einastofourierprofiletablemake
Modules used: iso_c_binding numerical_constants_math

**function: einastofreefallradius**
Description: Returns the freefall radius in the Einasto density profile at the specified time (given in Gyr).
Code lines: 57
Contained by: file dark_matter_profiles.Einasto.F90
Modules used: iso_c_binding numerical_constants_astronomical numerical_constants_physical numerical_interpolation

**function: einastofreefallradiusincreaserate**
Description: Returns the rate of increase of the freefall radius in the Einasto density profile at the specified time (given in Gyr).
Code lines: 58
Contained by: file dark_matter_profiles.Einasto.F90
Modules used: iso_c_binding numerical_constants_astronomical numerical_constants_physical numerical_interpolation

**subroutine: einastofreefalltabulate**
Description: Tabulates the freefall time vs. freefall radius for Einasto halos.
Code lines: 80
Contained by: file dark_matter_profiles.Einasto.F90
Modules used: galacticus_display memory_management numerical_interpolation numerical_ranges

**function: einastofreefalltimescalefree**
Description: Compute the freefall time in a scale-free Einasto halo.
19. Source Code Documentation

**Code lines:** 32  
**Contained by:** file `dark_matter_profiles.Einasto.F90`  
**Modules used:** `iso_c_binding`  

**function: einastofreefalltimescalefreeintegrand**  
**Description:** Integrand function used for finding the free-fall time in Einasto halos.  
**Code lines:** 10  
**Contained by:** function `einastofreefalltimescalefree`  
**Modules used:** `iso_c_binding`  

**function: einastokspace**  
**Description:** Returns the Fourier transform of the Einasto density profile at the specified waveNumber (given in Mpc\(^{-1}\)).  
**Code lines:** 45  
**Contained by:** file `dark_matter_profiles.Einasto.F90`  
**Modules used:** `dark_matter_halo_scales`  

**function: einastopotential**  
**Description:** Returns the potential (in (km/s)^2) in the dark matter profile of node at the given radius (given in units of Mpc).  
**Code lines:** 26  
**Contained by:** file `dark_matter_profiles.Einasto.F90`  
**Modules used:** `dark_matter_halo_scales`  

**function: einastopotentialscalefree**  
**Description:** Returns the gravitational potential (in units where the virial mass and scale radius are unity) in an Einasto dark matter profile with given concentration and alpha at the given radius (given in units of the scale radius).  
**Code lines:** 15  
**Contained by:** file `dark_matter_profiles.Einasto.F90`  
**Modules used:** `gamma_functions`  

**function: einastoradiusfromspecificangularmomentum**  
**Description:** Returns the radius (in Mpc) in node at which a circular orbit has the given specificAngularMomentum (given in units of km s\(^{-1}\) Mpc).  
**Code lines:** 22  
**Contained by:** file `dark_matter_profiles.Einasto.F90`  
**Modules used:** `numerical_constants_physical`  

**function: einastoradiusfromspecificangularmomentumscalefree**  
**Description:** Comptue the radius at which a circular orbit has the given specificAngularMomentumScaleFree in a scale free Einasto profile.  
**Code lines:** 32  
**Contained by:** file `dark_matter_profiles.Einasto.F90`  
**Modules used:** `iso_c_binding`  

**subroutine: einastoradiusfromspecificangularmomentumtablemake**
19.1. Program units

**Description:** Create a tabulation of the relation between specific angular momentum and radius in an Einasto profile.

**Code lines:** 69

**Contained by:** file `dark_matter_profiles.Einasto.F90`

**Modules used:**
- gamma_functions
- numerical_interpolation
- memory_management
- numerical_ranges

**function: einastorotationnormalization**

**Description:** Return the rotation normalization of an Einasto halo density profile.

**Code lines:** 23

**Contained by:** file `dark_matter_profiles.Einasto.F90`

**Modules used:**
- dark_matter_halo_scales
- gamma_functions
- numerical_constants_math

**subroutine: einastostaterestore**

**Description:** Retrieve the tabulation state from the file.

**Code lines:** 15

**Contained by:** file `dark_matter_profiles.Einasto.F90`

**subroutine: einastostatestore**

**Description:** Write the tabulation state to file.

**Code lines:** 10

**Contained by:** file `dark_matter_profiles.Einasto.F90`

**file: dark_matter_profiles.F90**

**Description:** Contains a module which provides an object that implements dark matter halo profiles.

**Code lines:** 3073

**module: dark_matter_profiles**

**Description:** Provides an object that implements dark matter halo profiles. An implementation of isothermal dark matter halo profiles. An implementation of “Einasto” dark matter halo profiles. An implementation of Navarro et al. [1997] dark matter halo profiles.

**Code lines:** 3053

**Contained by:** file `dark_matter_profiles.F90`

**Modules used:**
- dark_matter_halo_scales
- fgsl
- galacticus_nodes
- iso_varying_string
- kind_numbers
- tables

**Used by:**
- function `freefall_radius_dark_matter_halo`
- function `cooling_specific-angular_momentum_constant_rotation`
- function `dark_matter_halo_angular_momentum_growth_rate`
- function `dark_matter_profile_scale`
- subroutine `solve_for_radius`
- function `galactic_structure_radius_initial_adiabatic_solver`
- function `galactic_structure_radius_initial_derivative_adiabatic_solver`
- function `freefall_radius_growth_rate_dark_matter_halo`
- function `dark_matter_halo_angular_momentum`
- module `dark_matter_profiles_concentration`
- module `dark_matter_profiles_structure_tasks`
- subroutine `solve_for_radius`
subroutine galacticus_calculations_reset
subroutine galacticus_state_retrieve
subroutine halo_mass_function_compute
function half_mass_radius_root
function boylankolchin2008timeuntilmerging
function jiang2008timeuntilmerging
function extremum_solver

interface: darkmatterprofile
Code lines: 3
Contained by: module dark_matter_profiles

subroutine darkmatterprofilecalculationreset
Description: Reset the calculation state of the object.
Code lines: 7
Contained by: module dark_matter_profiles

function darkmatterprofilecircularvelocitymaximumnull
Description: Returns the maximum circular velocity (in km/s) in the dark matter profile of node.
Code lines: 8
Contained by: module dark_matter_profiles
Modules used: galacticus_error

function darkmatterprofilecircularvelocitynull
Description: Returns the circular velocity (in km/s) in the dark matter profile of node at the given radius (given in units of Mpc).
Code lines: 9
Contained by: module dark_matter_profiles
Modules used: galacticus_error

type: darkmatterprofileclass
Code lines: 119
Contained by: module dark_matter_profiles

function darkmatterprofileconstructordefault
Description: Return a pointer to the default darkMatterProfile object.
Code lines: 8
Contained by: module dark_matter_profiles

function darkmatterprofileconstructornamed
Description: Return a pointer to a newly created darkMatterProfile object of the specified type.
Code lines: 37
Contained by: module dark_matter_profiles
Modules used: galacticus_error iso_varying_string
function: darkmatterprofiledensitynull
  Description: Returns the density (in $M_\odot \text{Mpc}^{-3}$) in the dark matter profile of node at the given radius (given in units of Mpc).
  Code lines: 9
  Contained by: module dark_matter_profiles
  Modules used: galacticus_error

subroutine: darkmatterprofiledocalculationreset
  Description: Store the state to file.
  Code lines: 9
  Contained by: module dark_matter_profiles

subroutine: darkmatterprofiledostateretrieve
  Description: Retrieve the state from file.
  Code lines: 10
  Contained by: module dark_matter_profiles

subroutine: darkmatterprofiledostatestore
  Description: Store the state to file.
  Code lines: 10
  Contained by: module dark_matter_profiles

interface: darkmatterprofileeinasto
  Description: Constructors for the einasto dark matter halo profile class.
  Code lines: 4
  Contained by: module dark_matter_profiles

function: darkmatterprofileenclosedmassnull
  Description: Returns the enclosed mass (in $M_\odot$) in the dark matter profile of node at the given radius (given in units of Mpc). for the given node.
  Code lines: 9
  Contained by: module dark_matter_profiles
  Modules used: galacticus_error

function: darkmatterprofileenergygrowthratenull
  Description: Returns the rate of chance of the total energy of node in units of $M_\odot \text{km}^2 \text{s}^{-1} \text{Gyr}^{-1}$.
  Code lines: 8
  Contained by: module dark_matter_profiles
  Modules used: galacticus_error

function: darkmatterprofileenergynull
  Description: Return the total energy for the given node in units of $M_\odot \text{km}^2 \text{s}^{-1}$.
  Code lines: 8
  Contained by: module dark_matter_profiles
  Modules used: galacticus_error

function: darkmatterprofilefreefallradiusincreaseratenull
  Description: Returns the rate of increase of the freefall radius (in Mpc/Gyr) corresponding to the given time (in Gyr) in node.
19. Source Code Documentation

Code lines: 9
Contained by: module dark_matter_profiles
Modules used: galacticus_error

function: darkmatterprofilefreefallradiusnull
Description: Returns the freefall radius (in Mpc) corresponding to the given time (in Gyr) in node.
Code lines: 9
Contained by: module dark_matter_profiles
Modules used: galacticus_error

subroutine: darkmatterprofileinitialize
Description: Initialize the default darkMatterProfile object.
Code lines: 54
Contained by: module dark_matter_profiles
Modules used: galacticus_error, input_parameters, iso_varying_string

function: darkmatterprofileisfinalizable
Description: Return true if this object can be finalized.
Code lines: 7
Contained by: module dark_matter_profiles

interface: darkmatterprofileisothermal
Description: Constructors for the isothermal dark matter halo profile class.
Code lines: 4
Contained by: module dark_matter_profiles

function: darkmatterprofilekspacenull
Description: Returns the normalized Fourier space density profile of the dark matter profile of node at the given waveNumber (given in units of Mpc$^{-1}$).
Code lines: 9
Contained by: module dark_matter_profiles
Modules used: galacticus_error

interface: darkmatterprofilenfw
Description: Constructors for the nfw dark matter halo profile class.
Code lines: 4
Contained by: module dark_matter_profiles

function: darkmatterprofilepotentialnull
Description: Returns the gravitational potential (in (km/s)$^2$) in the dark matter profile of node at the given radius (given in units of Mpc).
Code lines: 10
Contained by: module dark_matter_profiles
Modules used: galacticus_error

function: darkmatterprofileradiusfromspecificangularmomentumnull
Description: Returns the radius (in Mpc) in the dark matter profile of node at which the specific angular momentum of a circular orbit equals specificAngularMomentum (specified in units of km s$^{-1}$ Mpc).
Code lines: 9
19.1. Program units

**Contained by:** module dark_matter_profiles
**Modules used:** galacticus_error

**function:** darkmatterprofilerotationnormalizationnull
**Description:** Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for the given node. Specifically, the normalization, \( A \), returned is such that \( V_{\text{rot}} = AJ/M \)
**Code lines:** 8

**Subroutine:** darkmatterprofilestaterestore
**Description:** Restore the state of the object to file.
**Code lines:** 9

**Subroutine:** darkmatterprofilestatestore
**Description:** Store the state of the object to file.
**Code lines:** 9

**Subroutine:** einastocalculationreset
**Description:** Reset the dark matter profile calculation.
**Code lines:** 8

**Function:** einastocircularvelocity
**Description:** Returns the circular velocity (in km/s) in the dark matter profile of node at the given radius (given in units of Mpc).
**Code lines:** 15

**Function:** einastocircularvelocitymaximum
**Description:** Returns the maximum circular velocity (in km/s) in the dark matter profile of node.
**Code lines:** 37

**Function:** einastocircularvelocitiespeakradius
**Description:** Computes the derivative of the square of circular velocity for an Einasto density profile.
**Code lines:** 8

**Function:** einastoconstructor
**Description:** Generic constructor for the einasto dark matter halo profile class.
19. Source Code Documentation

**function: einastodensity**

*Description:* Returns the density (in $M_\odot \text{Mpc}^{-3}$) in the dark matter profile of node at the given radius (given in units of Mpc).

*Code lines:* 21

*Contained by:* module dark_matter_profiles

*Modules used:* dark_matter_halo_scales

**function: einastodensityscalefree**

*Description:* Returns the density (in units such that the virial mass and scale length are unity) in an Einasto dark matter profile with given concentration and alpha at the given radius (given in units of the scale radius).

*Code lines:* 13

*Contained by:* module dark_matter_profiles

*Modules used:* gamma_functions, numerical_constants_math

**subroutine: einastodestructor**

*Description:* Destructor for the einasto dark matter halo profile class.

*Code lines:* 17

*Contained by:* module dark_matter_profiles

*Modules used:* numerical_interpolation

**function: einastoenclosedmass**

*Description:* Returns the enclosed mass (in $M_\odot$) in the dark matter profile of node at the given radius (given in units of Mpc).

*Code lines:* 21

*Contained by:* module dark_matter_profiles

*Modules used:* dark_matter_halo_scales

**function: einastoenclosedmassscalefree**

*Description:* Returns the enclosed mass (in units of the virial mass) in an Einasto dark matter profile with given concentration at the given radius (given in units of the scale radius).

*Code lines:* 14

*Contained by:* module dark_matter_profiles

*Modules used:* gamma_functions

**function: einastoenergy**

*Description:* Return the energy of an Einasto halo density profile.

*Code lines:* 41

*Contained by:* module dark_matter_profiles

*Modules used:* dark_matter_halo_scales, iso_c_binding
19.1. Program units

**numerical_interpolation**

**function: einastoenergygrowthrate**

*Description:* Return the energy of an Einasto halo density profile.

*Code lines:* 44

*Contained by:* module `dark_matter_profiles`

*Modules used:* `dark_matter_halo_scales` `iso_c_binding` `numerical_interpolation`

**subroutine: einastoenergytablemake**

*Description:* Create a tabulation of the energy of Einasto profiles as a function of their concentration of $\alpha$ parameter.

*Code lines:* 134

*Contained by:* module `dark_matter_profiles`

*Modules used:* `iso_c_binding` `memory_management` `numerical_constants_math` `numerical_integration` `numerical_interpolation` `numerical_ranges`

**function: einastojeansequationintegrand**

*Description:* Integrand for Einasto profile Jeans equation.

*Code lines:* 10

*Contained by:* subroutine `einastoenergytablemake`

*Modules used:* `iso_c_binding`

**function: einastokineticenergyintegrand**

*Description:* Integrand for Einasto profile kinetic energy.

*Code lines:* 10

*Contained by:* subroutine `einastoenergytablemake`

*Modules used:* `iso_c_binding`

**function: einastopotentialenergyintegrand**

*Description:* Integrand for Einasto profile potential energy.

*Code lines:* 10

*Contained by:* subroutine `einastoenergytablemake`

*Modules used:* `iso_c_binding`

**subroutine: einastofourierprofiletablemake**

*Description:* Create a tabulation of the Fourier transform of Einasto profiles as a function of their $\alpha$ parameter and dimensionless wavenumber.

*Code lines:* 117

*Contained by:* module `dark_matter_profiles`

*Modules used:* `galacticus_display` `iso_c_binding` `memory_management` `numerical_integration` `numerical_interpolation` `numerical_ranges`

**function: einastofourierprofileintegrand**

*Description:* Integrand for Einasto Fourier profile.

*Code lines:* 11

*Contained by:* subroutine `einastofourierprofiletablemake`
Module: einastofreefallradius
Description: Returns the freefall radius in the Einasto density profile at the specified time (given in Gyr).
Code lines: 57
Modules used: iso_c_binding, numerical_constants_math

Function: einastofreefallradiusincreaserate
Description: Returns the rate of increase of the freefall radius in the Einasto density profile at the specified time (given in Gyr).
Code lines: 58
Modules used: iso_c_binding, numerical_constants_astronomical, numerical_constants_physical, numerical_interpolation

Subroutine: einastofreefalltabulate
Description: Tabulates the freefall time vs. freefall radius for Einasto halos.
Code lines: 80
Modules used: galacticus_display, memory_management, numerical_interpolation, numerical_ranges

Function: einastofreefalltimescalefree
Description: Computes the freefall time in a scale-free Einasto halo.
Code lines: 32
Modules used: iso_c_binding, numerical_integration

Function: einastofreefalltimescalefreeintegrand
Description: Integrand function used for finding the free-fall time in Einasto halos.
Code lines: 10
Modules used: iso_c_binding

Function: einastokspace
Description: Returns the Fourier transform of the Einasto density profile at the specified waveNumber (given in Mpc⁻¹).
Code lines: 45
Modules used: dark_matter_halo_scales, iso_c_binding, numerical_interpolation

Function: einastopotential
Description: Returns the potential (in (km/s)^2) in the dark matter profile of node at the given radius (given in units of Mpc).
Code lines: 26
Modules used: dark_matter_halo_scales, dark_matter_profiles_error_codes
19.1. Program units

function: einastopotentialscalefree
Description: Returns the gravitational potential (in units where the virial mass and scale radius are unity) in an Einasto dark matter profile with given concentration and alpha at the given radius (given in units of the scale radius).
Code lines: 15
Contained by: module dark_matter_profiles
Modules used: gamma_functions

function: einastoradiusfromspecificangularmomentum
Description: Returns the radius (in Mpc) in node at which a circular orbit has the given specificAngularMomentum (given in units of km s\(^{-1}\) Mpc).
Code lines: 22
Contained by: module dark_matter_profiles
Modules used: numerical_constants_physical

function: einastoradiusfromspecificangularmomentumscalefree
Description: Comptue the radius at which a circular orbit has the given specificAngularMomentumScaleFree in a scale free Einasto profile.
Code lines: 32
Contained by: module dark_matter_profiles
Modules used: iso_c_binding numerical_interpolation

subroutine: einastoradiusfromspecificangularmomentumtablemake
Description: Create a tabulation of the relation between specific angular momentum and radius in an Einasto profile.
Code lines: 69
Contained by: module dark_matter_profiles
Modules used: gamma_functions memory_management numerical_interpolation numerical_ranges

function: einastorotationnormalization
Description: Return the rotation normalization of an Einasto halo density profile.
Code lines: 23
Contained by: module dark_matter_profiles
Modules used: dark_matter_halo_scales gamma_functions numerical_constants_math

subroutine: einastostaterestore
Description: Retrieve the tabulation state from the file.
Code lines: 15
Contained by: module dark_matter_profiles

subroutine: einastostatestore
Description: Write the tabulation state to file.
Code lines: 10
Contained by: module dark_matter_profiles
source Code Documentation

subroutine: isothermalcalculationreset
Description: Reset the dark matter profile calculation.
Code lines: 8
Contained by: module dark_matter_profiles

function: isothermalcircularvelocity
Description: Returns the circular velocity (in km/s) in the dark matter profile of node at the given radius (given in units of Mpc). For an isothermal halo this is independent of radius and therefore equal to the virial velocity.
Code lines: 12
Contained by: module dark_matter_profiles
Modules used: dark_matter_halo_scales galacticus_nodes

function: isothermalcircularvelocitymaximum
Description: Returns the maximum circular velocity (in km/s) in the dark matter profile of node. For an isothermal halo circular velocity is independent of radius.
Code lines: 11
Contained by: module dark_matter_profiles
Modules used: dark_matter_halo_scales galacticus_nodes

function: isothermalconstructor
Description: Generic constructor for the isothermal dark matter halo profile class.
Code lines: 9
Contained by: module dark_matter_profiles
Modules used: input_parameters

function: isothermaldefaultconstructor
Description: Default constructor for the isothermal dark matter halo profile class.
Code lines: 8
Contained by: module dark_matter_profiles
Modules used: input_parameters

function: isothermaldensity
Description: Returns the density (in $M_{\odot} \text{Mpc}^{-3}$) in the dark matter profile of node at the given radius (given in units of Mpc).
Code lines: 15
Contained by: module dark_matter_profiles
Modules used: dark_matter_halo_scales galacticus_nodes numerical_constants_math

subroutine: isothermaldestructor
Description: Destructor for the isothermal dark matter halo profile class.
Code lines: 7
Contained by: module dark_matter_profiles

function: isothermalenclosedmass
Description: Returns the enclosed mass (in $M_{\odot}$) in the dark matter profile of node at the given radius (given in units of Mpc).
Code lines: 14
Contained by: module dark_matter_profiles
19.1. Program units

**Function: isothermalenergy**

*Description:* Return the energy of an isothermal halo density profile.

*Code lines:* 12

*Contained by:* module `dark_matter_profiles`

*Modules used:* `dark_matter_halo_scales` `galacticus_nodes`

**Function: isothermalenergygrowthrate**

*Description:* Return the rate of change of the energy of an isothermal halo density profile.

*Code lines:* 12

*Contained by:* module `dark_matter_profiles`

*Modules used:* `dark_matter_halo_scales` `galacticus_nodes`

**Function: isothermalfreefallradius**

*Description:* Returns the freefall radius in the isothermal density profile at the specified time (given in Gyr). For an isothermal potential, the freefall radius, $r_{ff}(t)$, is:

$$ r_{ff}(t) = \sqrt{\frac{2}{\pi}} V_{\text{virial}} t. $$  \hspace{1cm} (19.3)

*Code lines:* 16

*Contained by:* module `dark_matter_profiles`

*Modules used:* `dark_matter_halo_scales` `galacticus_nodes`

**Function: isothermalfreefallradiusincreaserate**

*Description:* Returns the rate of increase of the freefall radius in the isothermal density profile at the specified time (given in Gyr). For an isothermal potential, the rate of increase of the freefall radius, $\dot{r}_{ff}(t)$, is:

$$ \dot{r}_{ff}(t) = \sqrt{\frac{2}{\pi}} V_{\text{virial}}. $$  \hspace{1cm} (19.4)

*Code lines:* 16

*Contained by:* module `dark_matter_profiles`

*Modules used:* `dark_matter_halo_scales` `galacticus_nodes`

**Function: isothermalkspace**

*Description:* Returns the Fourier transform of the isothermal density profile at the specified waveNumber (given in Mpc$^{-1}$), using the expression given in Cooray and Sheth (2002; table 1).

*Code lines:* 22

*Contained by:* module `dark_matter_profiles`

*Modules used:* `dark_matter_halo_scales` `galacticus_nodes` `numerical_constants_astronomical`

**Function: isothermalpotential**

*Description:* Returns the potential (in (km/s)$^2$) in the dark matter profile of node at the given radius (given in units of Mpc).

*Code lines:* 24

*Contained by:* module `dark_matter_profiles`

*Modules used:* `dark_matter_halo_scales` `dark_matter_profiles_error_codes`
function: isothermalradiusfromspecificangularmomentum

Description: Returns the radius (in Mpc) in node at which a circular orbit has the given specificAngularMomentum (given in units of km s\(^{-1}\) Mpc). For an isothermal halo, the circular velocity is constant (and therefore equal to the virial velocity). Therefore, \( r = \frac{j}{V_{\text{virial}}} \)
where \( j(=\text{specificAngularMomentum}) \) is the specific angular momentum and \( r \) the required radius.

Code lines: 14

Modules used: dark_matter_profiles galacticus_nodes

function: isothermalrotationnormalization

Description: Return the normalization of the rotation velocity vs. specific angular momentum relation.

Code lines: 10

Modules used: dark_matter_profiles galacticus_nodes

function: nfwangularmomentumscalefree

Description: Returns the total angular momentum (in units of the virial mass times scale radius times [assumed constant] rotation speed) in an NFW dark matter profile with given concentration. This is given by:

\[
J = \int_{0}^{c} 4\pi x^3 \rho(x) dx / \int_{0}^{c} 4\pi x^2 \rho(x) dx,
\]
(19.5)

where \( x \) is radius in units of the scale radius and \( c \) is concentration. This can be evaluated to give

\[
J = \left[ 1 + c - 2 \ln(1 + c) - \frac{1}{1 + c} \right] / \left[ \ln(1 + c) - \frac{c}{1 + c} \right].
\]
(19.6)

Code lines: 16

Modules used: dark_matter_profiles

subroutine: nfwcalculationreset

Description: Reset the dark matter profile calculation.

Code lines: 10

Modules used: dark_matter_profiles

function: nfwcircularvelocity

Description: Returns the circular velocity (in km/s) in the dark matter profile of node at the given radius (given in units of Mpc).

Code lines: 15

Modules used: dark_matter_profiles numerical_constants_physical

function: nfwcircularvelocitymaximum

Description: Returns the maximum circular velocity (in km/s) in the dark matter profile of node.

Code lines: 15

Modules used: dark_matter_profiles numerical_constants_physical
function: nfwconstructor
Description: Generic constructor for the nfw dark matter halo profile class.
Code lines: 25
Contained by: module dark_matter_profiles
Modules used: galacticus_error

function: nfwdefaultconstructor
Description: Default constructor for the nfw dark matter halo profile class.
Code lines: 8
Contained by: module dark_matter_profiles
Modules used: input_parameters

function: nfwdensity
Description: Returns the density (in $M_\odot$ Mpc$^{-3}$) in the dark matter profile of node at the given radius (given in units of Mpc).
Code lines: 19
Contained by: module dark_matter_profiles
Modules used: dark_matter_halo_scales

function: nfwdensitiescalefree
Description: Returns the density (in units such that the virial mass and scale length are unity) in an NFW dark matter profile with given concentration at the given radius (given in units of the scale radius).
Code lines: 10
Contained by: module dark_matter_profiles
Modules used: numerical_constants_math

subroutine: nfwdestructor
Description: Destructor for the nfw dark matter halo profile class.
Code lines: 19
Contained by: module dark_matter_profiles

function: nfwenclosedmass
Description: Returns the enclosed mass (in $M_\odot$) in the dark matter profile of node at the given radius (given in units of Mpc).
Code lines: 19
Contained by: module dark_matter_profiles
Modules used: dark_matter_halo_scales

function: nfwenclosedmassscalefree
Description: Returns the enclosed mass (in units of the virial mass) in an NFW dark matter profile with given concentration at the given radius (given in units of the scale radius).
Code lines: 31
Contained by: module dark_matter_profiles

function: nfwenergy
Description: Return the energy of an NFW halo density profile.
Code lines: 23
Contained by: module dark_matter_profiles
19. Source Code Documentation

**Modules used:** dark_matter_halo_scales

**function:** nfwenergygrowthrate
**Description:** Return the rate of change of the energy of an NFW halo density profile.
**Code lines:** 27
**Contained by:** module dark_matter_profiles
**Modules used:** dark_matter_halo_scales

**function:** nfwfreefallradius
**Description:** Returns the freefall radius in the NFW density profile at the specified time (given in Gyr).
**Code lines:** 41
**Contained by:** module dark_matter_profiles
**Modules used:** dark_matter_halo_scales numerical_constants_astronomical

**function:** nfwfreefallradiusincreaserate
**Description:** Returns the rate of increase of the freefall radius in the NFW density profile at the specified time (given in Gyr).
**Code lines:** 42
**Contained by:** module dark_matter_profiles
**Modules used:** dark_matter_halo_scales numerical_constants_astronomical

**subroutine:** nfwfreefalltabulate
**Description:** Tabulates the freefall time vs. freefall radius for NFW halos.
**Code lines:** 40
**Contained by:** module dark_matter_profiles

**function:** nfwfreefalltimescalefree
**Description:** Compute the freefall time in a scale-free NFW halo.
**Code lines:** 54
**Contained by:** module dark_matter_profiles
**Modules used:** iso_c_binding numerical_integration

**function:** nfwfreefalltimescalefreeintegrand
**Description:** Integrand function used for finding the free-fall time in NFW halos.
**Code lines:** 24
**Contained by:** function nfwfreefalltimescalefree
**Modules used:** iso_c_binding

**subroutine:** nfwinverseangularmomentum
**Description:** Tabulates the specific angular momentum vs. radius in an NFW profile for rapid inversion.
**Code lines:** 42
**Contained by:** module dark_matter_profiles

**function:** nfwkspace
**Description:** Returns the Fourier transform of the NFW density profile at the specified waveNumber (given in Mpc⁻¹), using the expression given in Cooray and Sheth (2002; eqn. 81).
**Code lines:** 27
**Contained by:** module dark_matter_profiles
**Modules used:** dark_matter_halo_scales exponential_integrals
function: nfwpotential
Description: Returns the potential (in \(\text{km/s}^2\)) in the dark matter profile of node at the given radius (given in units of Mpc).
Code lines: 27
Contained by: module dark_matter_profiles
Modules used: dark_matter_halo_scales dark_matter_profiles_error_codes

function: nfwpotentialenergy
Description: Computes the total energy of an NFW profile halo of given concentration using the methods of Cole et al. (2000; their Appendix A).
Code lines: 76
Contained by: module dark_matter_profiles
Modules used: iso_c_binding numerical_constants_math numerical_integration

function: nfwjeansequationintegrand
Description: Integrand for NFW profile Jeans equation.
Code lines: 10
Contained by: function nfwpotentialenergy
Modules used: iso_c_binding

function: fwkineticenergyintegrand
Description: Integrand for NFW profile kinetic energy.
Code lines: 10
Contained by: function nfwpotentialenergy
Modules used: iso_c_binding

function: nfwpotentialenergyintegrand
Description: Integrand for NFW profile potential energy.
Code lines: 10
Contained by: function nfwpotentialenergy
Modules used: iso_c_binding

function: nfwradiusfromspecificangularmomentum
Description: Returns the radius (in Mpc) in node at which a circular orbit has the given specificAngularMomentum (given in units of km s\(^{-1}\) Mpc). For an NFW halo, the circular velocity is constant (and therefore equal to the virial velocity). Therefore, \(r = j/V_{\text{virial}}\) where \(j(=\text{specificAngularMomentum})\) is the specific angular momentum and \(r\) the required radius.
Code lines: 47
Contained by: module dark_matter_profiles

function: nfwrotationnormalization
Description: Return the normalization of the rotation velocity vs. specific angular momentum relation.
Code lines: 21
Contained by: module dark_matter_profiles
Modules used: dark_matter_halo_scales
function: nfwspecificangularmomentumscalefree
Description: Returns the specific angular momentum, normalized to unit scale length and unit velocity at the scale radius, at position radius (in units of the scale radius) in an NFW profile.
Code lines: 9
Contained by: module dark_matter_profiles

subroutine: nfustaterestore
Description: Retrieve the tabulation state from the file.
Code lines: 16
Contained by: module dark_matter_profiles

subroutine: nfustatestore
Description: Write the tabulation state to file.
Code lines: 9
Contained by: module dark_matter_profiles

subroutine: nfutabulate
Description: Tabulate properties of the NFW halo profile which must be computed numerically.
Code lines: 35
Contained by: module dark_matter_profiles

file: dark_matter_profiles.NFW.F90
Description: An implementation of Navarro et al. [1997] dark matter halo profiles.
Code lines: 930
Modules used: dark_matter_halo_scales     kind_numbers
tables

interface: darkmatterprofilenfw
Description: Constructors for the nfw dark matter halo profile class.
Code lines: 4
Contained by: file dark_matter_profiles.NFW.F90

function: nfwangularmomentumscalefree
Description: Returns the total angular momentum (in units of the virial mass times scale radius times assumed constant rotation speed) in an NFW dark matter profile with given concentration. This is given by:

\[ J = \int_0^c 4\pi x^3 \rho(x) dx / \int_0^c 4\pi x^2 \rho(x) dx, \]  

where \( x \) is radius in units of the scale radius and \( c \) is concentration. This can be evaluated to give

\[ J = \left[ 1 + c - 2 \ln(1 + c) - \frac{1}{1 + c} \right] / \left[ \ln(1 + c) - \frac{c}{1 + c} \right]. \]  

Code lines: 16
Contained by: file dark_matter_profiles.NFW.F90

subroutine: nfucalculationreset
Description: Reset the dark matter profile calculation.
Code lines: 10
19.1. Program units

**function: nfwcircularvelocity**

*Description:* Returns the circular velocity (in km/s) in the dark matter profile of node at the given radius (given in units of Mpc).

*Code lines:* 15

*Contained by:* file dark_matter_profiles.NFW.F90

*Modules used:* numerical_constants_physical

**function: nfwcircularvelocitymaximum**

*Description:* Returns the maximum circular velocity (in km/s) in the dark matter profile of node.

*Code lines:* 15

*Contained by:* file dark_matter_profiles.NFW.F90

*Modules used:* numerical_constants_physical

**function: nfwconstructor**

*Description:* Generic constructor for the nfw dark matter halo profile class.

*Code lines:* 25

*Contained by:* file dark_matter_profiles.NFW.F90

*Modules used:* galacticus_error

**function: nfwdefaultconstructor**

*Description:* Default constructor for the nfw dark matter halo profile class.

*Code lines:* 8

*Contained by:* file dark_matter_profiles.NFW.F90

*Modules used:* input_parameters

**function: nfwdensity**

*Description:* Returns the density (in $M_\odot \ Mpc^{-3}$) in the dark matter profile of node at the given radius (given in units of Mpc).

*Code lines:* 19

*Contained by:* file dark_matter_profiles.NFW.F90

*Modules used:* dark_matter_halo_scales

**function: nfwdensityscalefree**

*Description:* Returns the density (in units such that the virial mass and scale length are unity) in an NFW dark matter profile with given concentration at the given radius (given in units of the scale radius).

*Code lines:* 10

*Contained by:* file dark_matter_profiles.NFW.F90

*Modules used:* numerical_constants_math

**subroutine: nfwdestructor**

*Description:* Destructor for the nfw dark matter halo profile class.

*Code lines:* 19

*Contained by:* file dark_matter_profiles.NFW.F90

**function: nfwenclosedmass**

*Description:* Returns the enclosed mass (in $M_\odot$) in the dark matter profile of node at the given radius (given in units of Mpc).

*Code lines:* 19
19. Source Code Documentation

Contains by: file `dark_matter_profiles.NFW.F90`
Modules used: `dark_matter_halo_scales`

**function: nfwenclosedmassscalefree**
*Description:* Returns the enclosed mass (in units of the virial mass) in an NFW dark matter profile with given concentration at the given radius (given in units of the scale radius).
*Code lines:* 31
*Contained by:* file `dark_matter_profiles.NFW.F90`

**function: nfwenergy**
*Description:* Return the energy of an NFW halo density profile.
*Code lines:* 23
*Contained by:* file `dark_matter_profiles.NFW.F90`
*Modules used:* `dark_matter_halo_scales`

**function: nfwenergygrowthrate**
*Description:* Return the rate of change of the energy of an NFW halo density profile.
*Code lines:* 27
*Contained by:* file `dark_matter_profiles.NFW.F90`
*Modules used:* `dark_matter_halo_scales`

**function: nfwfreefallradius**
*Description:* Returns the freefall radius in the NFW density profile at the specified time (given in Gyr).
*Code lines:* 41
*Contained by:* file `dark_matter_profiles.NFW.F90`
*Modules used:* `dark_matter_halo_scales` `numerical_constants_astronomical`

**function: nfwfreefallradiusincreaserate**
*Description:* Returns the rate of increase of the freefall radius in the NFW density profile at the specified time (given in Gyr).
*Code lines:* 42
*Contained by:* file `dark_matter_profiles.NFW.F90`
*Modules used:* `dark_matter_halo_scales` `numerical_constants_astronomical`

**subroutine: nfwfreefalltabulate**
*Description:* Tabulates the freefall time vs. freefall radius for NFW halos.
*Code lines:* 40
*Contained by:* file `dark_matter_profiles.NFW.F90`

**function: nfwfreefalltimescalefree**
*Description:* Compute the freefall time in a scale-free NFW halo.
*Code lines:* 54
*Contained by:* file `dark_matter_profiles.NFW.F90`
*Modules used:* `iso_c_binding` `numerical_integration`

**function: nfwfreefalltimescalefreearintegrand**
*Description:* Integrand function used for finding the free-fall time in NFW halos.
*Code lines:* 24
*Contained by:* function `nfwfreefalltimescalefree`
*Modules used:* `iso_c_binding`
subroutine: nfwinverseangularmomentum
Description: Tabulates the specific angular momentum vs. radius in an NFW profile for rapid inversion.
Code lines: 42
Contained by: file dark_matter_profiles.NFW.F90

function: nfwkspace
Description: Returns the Fourier transform of the NFW density profile at the specified waveNumber (given in Mpc$^{-1}$), using the expression given in Cooray and Sheth (2002; eqn. 81).
Code lines: 27
Contained by: file dark_matter_profiles.NFW.F90
Modules used: dark_matter_halo_scales exponential_integrals

function: nfwpotential
Description: Returns the potential (in (km/s)$^2$) in the dark matter profile of node at the given radius (given in units of Mpc).
Code lines: 27
Contained by: file dark_matter_profiles.NFW.F90
Modules used: dark_matter_halo_scales dark_matter_profiles_error_codes

function: nfwprofileenergy
Description: Computes the total energy of an NFW profile halo of given concentration using the methods of Cole et al. (2000; their Appendix A).
Code lines: 76
Contained by: file dark_matter_profiles.NFW.F90
Modules used: iso_c_binding numerical_constants_math numerical_integration

function: nfwweansequationintegrand
Description: Integrand for NFW profile Jeans equation.
Code lines: 10
Contained by: function nfwprofileenergy
Modules used: iso_c_binding

function: nfwwkineticenergyintegrand
Description: Integrand for NFW profile kinetic energy.
Code lines: 10
Contained by: function nfwprofileenergy
Modules used: iso_c_binding

function: nfwwpotentialenergyintegrand
Description: Integrand for NFW profile potential energy.
Code lines: 10
Contained by: function nfwprofileenergy
Modules used: iso_c_binding

function: nfwwradiusfromspecificangularmomentum
**Description:** Returns the radius (in Mpc) in node at which a circular orbit has the given \texttt{specificAngularMomentum} (given in units of km s$^{-1}$ Mpc). For an NFW halo, the circular velocity is constant (and therefore equal to the virial velocity). Therefore, $r = j/V_{\text{virial}}$ where $j(=\texttt{specificAngularMomentum})$ is the specific angular momentum and $r$ the required radius.

**Code lines:** 47
19.1. Program units

**Contained by:** file *dark_matter_profiles.NFW.F90*

**function: nfwrotationnormalization**
*Description:* Return the normalization of the rotation velocity vs. specific angular momentum relation.
*Code lines:* 21
**Contained by:** file *dark_matter_profiles.NFW.F90*
**Modules used:** *dark_matter_halo_scales*

**function: nfwspecificangularmomentumscalefree**
*Description:* Returns the specific angular momentum, normalized to unit scale length and unit velocity at the scale radius, at position *radius* (in units of the scale radius) in an NFW profile.
*Code lines:* 9
**Contained by:** file *dark_matter_profiles.NFW.F90*

**subroutine: nfwstaterestore**
*Description:* Retrieve the tabulation state from the file.
*Code lines:* 16
**Contained by:** file *dark_matter_profiles.NFW.F90*

**subroutine: nfwstatestore**
*Description:* Write the tabulation state to file.
*Code lines:* 9
**Contained by:** file *dark_matter_profiles.NFW.F90*

**subroutine: nfwtabulate**
*Description:* Tabulate properties of the NFW halo profile which must be computed numerically.
*Code lines:* 35
**Contained by:** file *dark_matter_profiles.NFW.F90*

**file: dark_matter_profiles.error_codes.F90**
*Description:* Contains a module which provides error codes for dark matter profile calculations.
*Code lines:* 27

**module: dark_matter_profiles.error_codes**
*Description:* Provides error codes for dark matter profile calculations.
*Code lines:* 7
**Contained by:** file *dark_matter_profiles.error_codes.F90*
**Used by:**
- function *einstopotential*
- function *isothermalpotential*
- function *nfwpotential*
- function *nfwpotential scalefree*
- function *equivalent_circular_orbit_- solver*
- subroutine *satellite_orbit_extremum_- phase_space_coordinates*

**file: dark_matter_profiles.isothermal.F90**
*Description:* An implementation of isothermal dark matter halo profiles.
*Code lines:* 295
**Modules used:** *dark_matter_halo_scales*

**interface: darkmatterprofileisothermal**
Constructors for the isothermal dark matter halo profile class.

Description:
Reset the dark matter profile calculation.

Code lines: 8
Contained by: file dark_matter_profiles.isothermal.F90

Function: isothermalcircularvelocity
Description: Returns the circular velocity (in km/s) in the dark matter profile of node at the given radius (given in units of Mpc). For an isothermal halo this is independent of radius and therefore equal to the virial velocity.

Code lines: 12
Contained by: file dark_matter_profiles.isothermal.F90
Modules used: dark_matter_halo_scales galacticus_nodes

Function: isothermalcircularvelocitymaximum
Description: Returns the maximum circular velocity (in km/s) in the dark matter profile of node. For an isothermal halo circular velocity is independent of radius.

Code lines: 11
Contained by: file dark_matter_profiles.isothermal.F90
Modules used: dark_matter_halo_scales galacticus_nodes

Function: isothermalconstructor
Description: Generic constructor for the isothermal dark matter halo profile class.

Code lines: 9
Contained by: file dark_matter_profiles.isothermal.F90
Modules used: input_parameters

Function: isothermaldefaultconstructor
Description: Default constructor for the isothermal dark matter halo profile class.

Code lines: 8
Contained by: file dark_matter_profiles.isothermal.F90
Modules used: input_parameters

Function: isothermaldensity
Description: Returns the density (in $M_{\odot} \text{Mpc}^{-3}$) in the dark matter profile of node at the given radius (given in units of Mpc).

Code lines: 15
Contained by: file dark_matter_profiles.isothermal.F90
Modules used: dark_matter_halo_scales galacticus_nodes numerical_constants_math

Subroutine: isothermaldestructor
Description: Destructor for the isothermal dark matter halo profile class.

Code lines: 7
Contained by: file dark_matter_profiles.isothermal.F90
function: isothermalenclosedmass
Description: Returns the enclosed mass (in $M_\odot$) in the dark matter profile of node at the given radius (given in units of Mpc).
Code lines: 14
Contained by: file dark_matter_profiles.isothermal.F90
Modules used: dark_matter_halo_scales galacticus_nodes

function: isothermalenergy
Description: Return the energy of an isothermal halo density profile.
Code lines: 12
Contained by: file dark_matter_profiles.isothermal.F90
Modules used: dark_matter_halo_scales galacticus_nodes

function: isothermalenergygrowthrate
Description: Return the rate of change of the energy of an isothermal halo density profile.
Code lines: 12
Contained by: file dark_matter_profiles.isothermal.F90
Modules used: dark_matter_halo_scales galacticus_nodes

function: isothermalfreefallradius
Description: Returns the freefall radius in the isothermal density profile at the specified time (given in Gyr). For an isothermal potential, the freefall radius, $r_{ff}(t)$, is:

$$r_{ff}(t) = \sqrt{\frac{2}{\pi}} V_{\text{vir}} t.$$  \hspace{1cm} (19.9)

Code lines: 16
Contained by: file dark_matter_profiles.isothermal.F90
Modules used: dark_matter_halo_scales galacticus_nodes numerical_constants_astronomical

function: isothermalfreefallradiusincreaserate
Description: Returns the rate of increase of the freefall radius in the isothermal density profile at the specified time (given in Gyr). For an isothermal potential, the rate of increase of the freefall radius, $\dot{r}_{ff}(t)$, is:

$$\dot{r}_{ff}(t) = \sqrt{\frac{2}{\pi}} V_{\text{vir}}.$$  \hspace{1cm} (19.10)

Code lines: 16
Contained by: file dark_matter_profiles.isothermal.F90
Modules used: dark_matter_halo_scales galacticus_nodes numerical_constants_astronomical

function: isothermalkspace
Description: Returns the Fourier transform of the isothermal density profile at the specified waveNumber (given in Mpc$^{-1}$), using the expression given in Cooray and Sheth (2002; table 1).
Code lines: 22
Contained by: file dark_matter_profiles.isothermal.F90
Modules used: dark_matter_halo_scales exponential_integrals galacticus_nodes
function: isothermalpotential  
Description: Returns the potential (in (km/s)^2) in the dark matter profile of node at the given radius (given in units of Mpc).

Code lines: 24  
Contained by: file dark_matter_profiles.isothermal.F90  
Modules used: dark_matter_halo_scales dark_matter_profiles_error_codes galacticus_error galacticus_nodes

function: isothermalradiusfromspecificangularmomentum  
Description: Returns the radius (in Mpc) in node at which a circular orbit has the given specificAngularMomentum (given in units of km s^{-1} Mpc). For an isothermal halo, the circular velocity is constant (and therefore equal to the virial velocity). Therefore, \( r = \frac{j}{V_{\text{virial}}} \) where \( j (= \text{specificAngularMomentum}) \) is the specific angular momentum and \( r \) the required radius.

Code lines: 14  
Contained by: file dark_matter_profiles.isothermal.F90  
Modules used: dark_matter_halo_scales galacticus_nodes

function: isothermalrotationnormalization  
Description: Return the normalization of the rotation velocity vs. specific angular momentum relation.

Code lines: 10  
Contained by: file dark_matter_profiles.isothermal.F90  
Modules used: dark_matter_halo_scales galacticus_nodes


Code lines: 222

interface: darkmatterprofileconcentrationdiemerkravtsov2014  
Description: Constructors for the diemerKravtsov2014 dark matter halo profile concentration class.

Code lines: 4  

function: diemerkravtsov2014concentration  
Description: Return the concentration of the dark matter halo profile of node using the Diemer and Kravtsov [2014] algorithm.

Code lines: 23  
Modules used: cosmology_parameters critical_overdensity numerical_constants_math power_spectra

function: diemerkravtsov2014constructor  
Description: Constructor for the diemerKravtsov2014 dark matter halo profile concentration class.

Code lines: 15  
Modules used: galacticus_error
function: diemerkravtsov2014darkmatterprofiledefinition
Description: Return a dark matter density profile object defining that used in the definition of concentration in the Diemer and Kravtsov [2014] algorithm.
Code lines: 20
Modules used: dark_matter_halo_scales

function: diemerkravtsov2014defaultconstructor
Description: Default constructor for the diemerKravtsov2014 dark matter halo profile concentration class.
Code lines: 95
Modules used: input_parameters

function: diemerkravtsov2014densitycontrastdefinition
Description: Return a virial density contrast object defining that used in the definition of concentration in the Diemer and Kravtsov [2014] algorithm.
Code lines: 13

Description: An implementation of dark matter halo profile concentrations using the Dutton and Macciò [2014] algorithm.
Code lines: 207

interface: darkmatterprofileconcentrationduttonmaccio2014
Description: Constructors for the duttonMaccio2014 dark matter halo profile concentration class.
Code lines: 4

function: duttonmaccio2014concentration
Description: Return the concentration of the dark matter halo profile of node using the Dutton and Macciò [2014] algorithm.
Code lines: 23
Modules used: cosmology_functions

function: duttonmaccio2014constructor
Description: Constructor for the duttonMaccio2014 dark matter halo profile concentration class.
Code lines: 39
Modules used: galacticus_error

function: duttonmaccio2014darkmatterprofiledefinition
Description: Return a dark matter density profile object defining that used in the definition of concentration in the Diemer and Kravtsov [2014] algorithm.
Code lines: 30
Modules used: dark_matter_halo_scales
19. Source Code Documentation

**function:** duttonmaccio2014defaultconstructor

*Description:* Default constructor for the duttonMaccio2014 dark matter halo profile concentration class.

*Code lines:* 29


*Modules used:* input_parameters

**function:** duttonmaccio2014densitycontrastdefinition

*Description:* Return a virial density contrast object defining that used in the definition of concentration in the Dutton and Macciò [2014] algorithm.

*Code lines:* 21


**file:** dark_matter_profiles.structure.concentration.F90

*Description:* Contains a module which provides an object that implements concentrations of dark matter halo profiles.

*Code lines:* 1711

**module:** dark_matter_profiles_concentration


*Code lines:* 1691

*Contained by:* file dark_matter_profiles.structure.concentration.F90

*Modules used:* dark_matter_profiles galacticus_nodes iso_c_binding iso_varying_string virial_density_contrast

*Used by:* function dark_matter_profile_scale subroutine assign_scale_radii program test_nfw96_concentration dark_energy program test_zhao2009_flat program test_zhao2009_dark_energy program test_zhao2009_open

**interface:** darkmatterprofileconcentration

*Code lines:* 3

*Contained by:* module dark_matter_profiles_concentration

**type:** darkmatterprofileconcentrationclass

*Code lines:* 35

*Contained by:* module dark_matter_profiles_concentration
19.1. Program units

function: darkmatterprofileconcentrationconcentrationnull
Description: Returns the concentration parameter for the given node.
Code lines: 8
Contained by: module dark_matter_profiles_concentration
Modules used: galacticus_error

function: darkmatterprofileconcentrationconstructordefault
Description: Return a pointer to the default darkMatterProfileConcentration object.
Code lines: 8
Contained by: module dark_matter_profiles_concentration

function: darkmatterprofileconcentrationconstructornamed
Description: Return a pointer to a newly created darkMatterProfileConcentration object of the specified type.
Code lines: 72
Contained by: module dark_matter_profiles_concentration
Modules used: galacticus_error iso_varying_string

function: darkmatterprofileconcentrationdarkmatterprofiledefinitionnull
Description: Returns a darkMatterProfile object describing the dark matter density profile used to define this concentration.
Code lines: 8
Contained by: module dark_matter_profiles_concentration
Modules used: galacticus_error

function: darkmatterprofileconcentrationdensitycontrastdefinitionnull
Description: Returns a virialDensityContrast object describing the virial density contrast used to define this concentration.
Code lines: 8
Contained by: module dark_matter_profiles_concentration
Modules used: galacticus_error

interface: darkmatterprofileconcentrationdiemerkravtsov2014
Description: Constructors for the diemerKravtsov2014 dark matter halo profile concentration class.
Code lines: 4
Contained by: module dark_matter_profiles_concentration

interface: darkmatterprofileconcentrationduttonmaccio2014
Description: Constructors for the duttonMaccio2014 dark matter halo profile concentration class.
Code lines: 4
Contained by: module dark_matter_profiles_concentration

interface: darkmatterprofileconcentrationgao2008
Description: Constructors for the gao2008 dark matter halo profile concentration class.
Code lines: 3
Contained by: module dark_matter_profiles_concentration

subroutine: darkmatterprofileconcentrationinitialize
Description: Initialize the default darkMatterProfileConcentration object.
Code lines: 89
Contained by: module dark_matter_profiles_concentration
Modules used: galacticus_error input_parameters iso_varying_string

function: darkmatterprofileconcentrationisfinalizable
Description: Return true if this object can be finalized.
Code lines: 7
Contained by: module dark_matter_profiles_concentration

interface: darkmatterprofileconcentrationmunozcuartas2011
Description: Constructors for the munozCuartas2011 dark matter halo profile concentration class.
Code lines: 3
Contained by: module dark_matter_profiles_concentration

interface: darkmatterprofileconcentrationnfw1996
Description: Constructors for the nfw1996 dark matter halo profile concentration class.
Code lines: 4
Contained by: module dark_matter_profiles_concentration

interface: darkmatterprofileconcentrationprada2011
Description: Constructors for the prada2011 dark matter halo profile concentration class.
Code lines: 4
Contained by: module dark_matter_profiles_concentration

interface: darkmatterprofileconcentrationwdm
Description: Constructors for the WDM dark matter halo profile concentration class.
Code lines: 4
Contained by: module dark_matter_profiles_concentration

interface: darkmatterprofileconcentrationzhao2009
Description: Constructors for the zhao2009 dark matter halo profile concentration class.
Code lines: 3
Contained by: module dark_matter_profiles_concentration

function: diemerkravtsov2014concentration
Description: Return the concentration of the dark matter halo profile of node using the Diemer and Kravtsov [2014] algorithm.
Code lines: 23
Contained by: module dark_matter_profiles_concentration
Modules used: cosmology_parameters critical_overdensity numerical_constants_math power_spectra

function: diemerkravtsov2014constructor
Description: Constructor for the diemerKravtsov2014 dark matter halo profile concentration class.
Code lines: 15
Contained by: module dark_matter_profiles_concentration
Modules used: galacticus_error
function: diemerkravtsov2014darkmatterprofiledefinition
Description: Return a dark matter density profile object defining that used in the definition of concentration in the Diemer and Kravtsov [2014] algorithm.
Code lines: 20
Contained by: module dark_matter_profiles_concentration
Modules used: dark_matter_halo_scales

function: diemerkravtsov2014defaultconstructor
Description: Default constructor for the diemerKravtsov2014 dark matter halo profile concentration class.
Code lines: 95
Contained by: module dark_matter_profiles_concentration
Modules used: input_parameters

function: diemerkravtsov2014densitycontrastdefinition
Description: Return a virial density contrast object defining that used in the definition of concentration in the Diemer and Kravtsov [2014] algorithm.
Code lines: 13
Contained by: module dark_matter_profiles_concentration

function: duttonmaccio2014concentration
Description: Return the concentration of the dark matter halo profile of node using the Dutton and Macciò [2014] algorithm.
Code lines: 23
Contained by: module dark_matter_profiles_concentration
Modules used: cosmology_functions

function: duttonmaccio2014constructor
Description: Constructor for the duttonMaccio2014 dark matter halo profile concentration class.
Code lines: 39
Contained by: module dark_matter_profiles_concentration
Modules used: galacticus_error

function: duttonmaccio2014darkmatterprofiledefinition
Description: Return a dark matter density profile object defining that used in the definition of concentration in the Diemer and Kravtsov [2014] algorithm.
Code lines: 30
Contained by: module dark_matter_profiles_concentration
Modules used: dark_matter_halo_scales

function: duttonmaccio2014defaultconstructor
Description: Default constructor for the duttonMaccio2014 dark matter halo profile concentration class.
Code lines: 29
Contained by: module dark_matter_profiles_concentration
Modules used: input_parameters

function: duttonmaccio2014densitycontrastdefinition
Description: Return a virial density contrast object defining that used in the definition of concentration in the Dutton and Macciò [2014] algorithm.
Code lines: 21
function: gao2008concentration
Description: Return the concentration of the dark matter halo profile of node using the Gao et al. [2008] algorithm.
Code lines: 22
Contained by: module dark_matter_profiles_concentration

function: gao2008darkmatterprofiledefinition
Description: Return a dark matter density profile object defining that used in the definition of concentration in the Gao et al. [2008] algorithm.
Code lines: 20
Contained by: module dark_matter_profiles_concentration
Modules used: cosmology_functions

function: gao2008defaultconstructor
Description: Default constructor for the gao2008 dark matter halo profile concentration class.
Code lines: 6
Contained by: module dark_matter_profiles_concentration

function: gao2008densitycontrastdefinition
Description: Return a virial density contrast object defining that used in the definition of concentration in the Gao et al. [2008] algorithm.
Code lines: 12
Contained by: module dark_matter_profiles_concentration

function: munozcuartas2011concentration
Description: Return the concentration of the dark matter halo profile of node using the Muñoz-Cuartas et al. [2011] algorithm.
Code lines: 27
Contained by: module dark_matter_profiles_concentration
Modules used: cosmology_functions cosmology_parameters

function: munozcuartas2011darkmatterprofiledefinition
Description: Return a dark matter density profile object defining that used in the definition of concentration in the Muñoz-Cuartas et al. [2011] algorithm.
Code lines: 20
Contained by: module dark_matter_profiles_concentration
Modules used: dark_matter_halo_scales

function: munozcuartas2011defaultconstructor
Description: Default constructor for the munozCuartas2011 dark matter halo profile concentration class.
Code lines: 6
Contained by: module dark_matter_profiles_concentration

function: munozcuartas2011densitycontrastdefinition
Description: Return a virial density contrast object defining that used in the definition of concentration in the Muñoz-Cuartas et al. [2011] algorithm.
Code lines: 12
19.1. Program units

**Contained by:** module `dark_matter_profiles_concentration`

**function: nfw1996concentration**
*Description:* Return the concentration of the dark matter halo profile of node using the Navarro et al. [1996] algorithm.
*Code lines:* 47
*Contained by:* module `dark_matter_profiles_concentration`
*Modules used:* `cosmology_functions` `critical_overdensity` `power_spectra` `root_finder` `virial_density_contrast`

**function: nfw1996constructor**
*Description:* Constructor for the nfw1996 dark matter halo profile concentration class.
*Code lines:* 9
*Contained by:* module `dark_matter_profiles_concentration`

**function: nfw1996darkmatterprofiledefinition**
*Description:* Return a dark matter density profile object defining that used in the definition of concentration in the Navarro et al. [1996] algorithm.
*Code lines:* 20
*Contained by:* module `dark_matter_profiles_concentration`
*Modules used:* `dark_matter_halo_scales`

**function: nfw1996defaultconstructor**
*Description:* Default constructor for the nfw1996 dark matter halo profile concentration class.
*Code lines:* 40
*Contained by:* module `dark_matter_profiles_concentration`
*Modules used:* `input_parameters`

**function: nfw1996densitycontrastdefinition**
*Description:* Return a virial density contrast object defining that used in the definition of concentration in the Navarro et al. [1996] algorithm.
*Code lines:* 12
*Contained by:* module `dark_matter_profiles_concentration`

**function: nfw1996rootfunction**
*Description:* Root function used in finding concentrations in the Navarro et al. [1996] method.
*Code lines:* 7
*Contained by:* module `dark_matter_profiles_concentration`

**function: prada2011b0**
*Description:* The function $B_0(x)$ as defined in eqn. (18) of Prada et al. [2011].
*Code lines:* 7
*Contained by:* module `dark_matter_profiles_concentration`

**function: prada2011b1**
*Description:* The function $B_1(x)$ as defined in eqn. (18) of Prada et al. [2011].
*Code lines:* 7
*Contained by:* module `dark_matter_profiles_concentration`
function: prada2011c
Description: The function $C(\sigma')$ as defined in eqn. (17) of Prada et al. [2011].
Code lines: 7
Contained by: module dark_matter_profiles_concentration

function: prada2011cmin
Description: The function $c_{\text{min}}(x)$ as defined in eqn. (19) of Prada et al. [2011].
Code lines: 8
Contained by: module dark_matter_profiles_concentration
Modules used: numerical_constants_math

function: prada2011concentration
Description: Return the concentration of the dark matter halo profile of node using the Prada et al. [2011] algorithm.
Code lines: 25
Contained by: module dark_matter_profiles_concentration
Modules used: cosmology_functions cosmology_parameters linear_growth power_spectra

function: prada2011constructor
Description: Constructor for the prada2011 dark matter halo profile concentration class.
Code lines: 19
Contained by: module dark_matter_profiles_concentration

function: prada2011darkmatterprofiledefinition
Description: Return a dark matter density profile object defining that used in the definition of concentration in the Prada et al. [2011] algorithm.
Code lines: 20
Contained by: module dark_matter_profiles_concentration
Modules used: dark_matter_halo_scales

function: prada2011defaultconstructor
Description: Default constructor for the prada2011 dark matter halo profile concentration class.
Code lines: 150
Contained by: module dark_matter_profiles_concentration
Modules used: input_parameters

function: prada2011densitycontrastdefinition
Description: Return a virial density contrast object defining that used in the definition of concentration in the Prada et al. [2011] algorithm.
Code lines: 12
Contained by: module dark_matter_profiles_concentration

function: prada2011inversesigmamin
Description: The function $\sigma_{\text{min}}^{-1}(x)$ as defined in eqn. (20) of Prada et al. [2011].
Code lines: 8
Contained by: module dark_matter_profiles_concentration
Modules used: numerical_constants_math
function: wdmconcentration
Description: Return the concentration of the dark matter halo profile of node using the warm dark matter modifier of Schneider et al. [2012].
Code lines: 16
Contained by: module dark_matter_profiles_concentration
Modules used: transfer_functions

function: wdmdefaultconstructor
Description: Default constructor for the wdm dark matter halo profile concentration class.
Code lines: 30
Contained by: module dark_matter_profiles_concentration
Modules used: input_parameters

function: wdmgenericconstructor
Description: Generic constructor for the warm dark matter (WDM) dark matter halo concentration class.
Code lines: 9
Contained by: module dark_matter_profiles_concentration

function: zhao2009concentration
Description: Return the concentration of the dark matter halo profile of node using the Zhao et al. [2009] algorithm.
Code lines: 19
Contained by: module dark_matter_profiles_concentration
Modules used: dark_matter_halo_formation_times

function: zhao2009darkmatterprofiledefinition
Description: Return a dark matter density profile object defining that used in the definition of concentration in the Zhao et al. [2009] algorithm.
Code lines: 19
Contained by: module dark_matter_profiles_concentration
Modules used: dark_matter_halo_scales

function: zhao2009defaultconstructor
Description: Default constructor for the zhao2009 dark matter halo profile concentration class.
Code lines: 6
Contained by: module dark_matter_profiles_concentration
Modules used: input_parameters

function: zhao2009densitycontrastdefinition
Description: Return a virial density contrast object defining that used in the definition of concentration in the Zhao et al. [2009] algorithm.
Code lines: 12
Contained by: module dark_matter_profiles_concentration

Description: An implementation of dark matter halo profile concentrations using the Gao et al. [2008] algorithm.
Code lines: 107
interface: darkmatterprofileconcentrationgao2008
Description: Constructors for the gao2008 dark matter halo profile concentration class.
Code lines: 3

function: gao2008concentration
Description: Return the concentration of the dark matter halo profile of node using the Gao et al. [2008] algorithm.
Code lines: 22
Modules used: cosmology_functions

function: gao2008darkmatterprofiledefinition
Description: Return a dark matter density profile object defining that used in the definition of concentration in the Gao et al. [2008] algorithm.
Code lines: 20
Modules used: dark_matter_halo_scales

function: gao2008defaultconstructor
Description: Default constructor for the gao2008 dark matter halo profile concentration class.
Code lines: 6

function: gao2008densitycontrastdefinition
Description: Return a virial density contrast object defining that used in the definition of concentration in the Gao et al. [2008] algorithm.
Code lines: 12

Description: An implementation of dark matter halo profile concentrations using the Muñoz-Cuartas et al. [2011] algorithm.
Code lines: 112

interface: darkmatterprofileconcentrationmunozcuartas2011
Description: Constructors for the munozCuartas2011 dark matter halo profile concentration class.
Code lines: 3

function: munozcuartas2011concentration
Description: Return the concentration of the dark matter halo profile of node using the Muñoz-Cuartas et al. [2011] algorithm.
Code lines: 27
Modules used: cosmology_functions cosmology_parameters

function: munozcuartas2011darkmatterprofiledefinition
19.1. Program units

**Description:** Return a dark matter density profile object defining that used in the definition of concentration in the Muñoz-Cuartas et al. [2011] algorithm.

**Code lines:** 20

**Contained by:** file `dark_matter_profiles.structure.concentration.MunozCuartas2011.F90`

**Modules used:** `dark_matter_halo_scales`

**function:** `munozcuartas2011defaultconstructor`

**Description:** Default constructor for the `munozCuartas2011` dark matter halo profile concentration class.

**Code lines:** 6

**Contained by:** file `dark_matter_profiles.structure.concentration.MunozCuartas2011.F90`

**function:** `munozcuartas2011densitycontrastdefinition`

**Description:** Return a virial density contrast object defining that used in the definition of concentration in the Muñoz-Cuartas et al. [2011] algorithm.

**Code lines:** 12

**Contained by:** file `dark_matter_profiles.structure.concentration.MunozCuartas2011.F90`

**file:** `dark_matter_profiles.structure.concentration.NFW.F90`

**Description:** An implementation of dark matter halo profile concentrations using the Navarro et al. [1996] algorithm.

**Code lines:** 195

**interface:** `darkmatterprofileconcentrationnfw1996`

**Description:** Constructors for the `nfw1996` dark matter halo profile concentration class.

**Code lines:** 4

**Contained by:** file `dark_matter_profiles.structure.concentration.NFW.F90`

**function:** `nfw1996concentration`

**Description:** Return the concentration of the dark matter halo profile of node using the Navarro et al. [1996] algorithm.

**Code lines:** 47

**Contained by:** file `dark_matter_profiles.structure.concentration.NFW.F90`

**Modules used:** `cosmology_functions`  `critical_overdensity`

`power_spectra`  `root_finder`

`virial_density_contrast`

**function:** `nfw1996constructor`

**Description:** Constructor for the `nfw1996` dark matter halo profile concentration class.

**Code lines:** 9

**Contained by:** file `dark_matter_profiles.structure.concentration.NFW.F90`

**function:** `nfw1996darkmatterprofiledefinition`

**Description:** Return a dark matter density profile object defining that used in the definition of concentration in the Navarro et al. [1996] algorithm.

**Code lines:** 20

**Contained by:** file `dark_matter_profiles.structure.concentration.NFW.F90`

**Modules used:** `dark_matter_halo_scales`

**function:** `nfw1996defaultconstructor`

**Description:** Default constructor for the `nfw1996` dark matter halo profile concentration class.
19. Source Code Documentation

**Code lines:** 40  
**Contained by:** file `dark_matter_profiles.structure.concentration.NFW.F90`  
**Modules used:** `input_parameters`

**function:** nfw1996densitycontrastdefinition  
**Description:** Return a virial density contrast object defining that used in the definition of concentration in the Navarro et al. [1996] algorithm.  
**Code lines:** 12  
**Contained by:** file `dark_matter_profiles.structure.concentration.NFW.F90`

**function:** nfw1996rootfunction  
**Description:** Root function used in finding concentrations in the Navarro et al. [1996] method.  
**Code lines:** 7  
**Contained by:** file `dark_matter_profiles.structure.concentration.NFW.F90`

**file:** `dark_matter_profiles.structure.concentration.Prada2011.F90`  
**Description:** An implementation of dark matter halo profile concentrations using the Prada et al. [2011] algorithm.  
**Code lines:** 327

**interface:** darkmatterprofileconcentrationprada2011  
**Description:** Constructors for the prada2011 dark matter halo profile concentration class.  
**Code lines:** 4  
**Contained by:** file `dark_matter_profiles.structure.concentration.Prada2011.F90`

**function:** prada2011b0  
**Description:** The function $B_0(x)$ as defined in eqn. (18) of Prada et al. [2011].  
**Code lines:** 7  
**Contained by:** file `dark_matter_profiles.structure.concentration.Prada2011.F90`

**function:** prada2011b1  
**Description:** The function $B_1(x)$ as defined in eqn. (18) of Prada et al. [2011].  
**Code lines:** 7  
**Contained by:** file `dark_matter_profiles.structure.concentration.Prada2011.F90`

**function:** prada2011c  
**Description:** The function $C(\sigma')$ as defined in eqn. (17) of Prada et al. [2011].  
**Code lines:** 7  
**Contained by:** file `dark_matter_profiles.structure.concentration.Prada2011.F90`

**function:** prada2011cmin  
**Description:** The function $c_{\text{min}}(x)$ as defined in eqn. (19) of Prada et al. [2011].  
**Code lines:** 8  
**Contained by:** file `dark_matter_profiles.structure.concentration.Prada2011.F90`  
**Modules used:** `numerical_constants_math`

**function:** prada2011concentration  
**Description:** Return the concentration of the dark matter halo profile of `node` using the Prada et al. [2011] algorithm.  
**Code lines:** 25

1060
function: prada2011constructor
Description: Constructor for the prada2011 dark matter halo profile concentration class.
Code lines: 19
Contained by: file dark_matter_profiles.structure.concentration.Prada2011.F90

function: prada2011darkmatterprofiledefinition
Description: Return a dark matter density profile object defining that used in the definition of concentration in the Prada et al. [2011] algorithm.
Code lines: 20
Contained by: file dark_matter_profiles.structure.concentration.Prada2011.F90
Modules used: dark_matter_halo_scales

function: prada2011defaultconstructor
Description: Default constructor for the prada2011 dark matter halo profile concentration class.
Code lines: 150
Contained by: file dark_matter_profiles.structure.concentration.Prada2011.F90
Modules used: input_parameters

function: prada2011densitycontrastdefinition
Description: Return a virial density contrast object defining that used in the definition of concentration in the Prada et al. [2011] algorithm.
Code lines: 12
Contained by: file dark_matter_profiles.structure.concentration.Prada2011.F90

function: prada2011inversesigmamin
Description: The function $\sigma^{-1}_m(x)$ as defined in eqn. (20) of Prada et al. [2011].
Code lines: 8
Contained by: file dark_matter_profiles.structure.concentration.Prada2011.F90
Modules used: numerical_constants_math

file: dark_matter_profiles.structure.concentration.WDM.F90
Description: An implementation of warm dark matter halo profile concentrations using the Schneider et al. [2012] modifier.
Code lines: 107

interface: darkmatterprofileconcentrationwdm
Description: Constructors for the WDM dark matter halo profile concentration class.
Code lines: 4
Contained by: file dark_matter_profiles.structure.concentration.WDM.F90

function: wdmconcentration
Description: Return the concentration of the dark matter halo profile of node using the warm dark matter modifier of Schneider et al. [2012].
Code lines: 16
Contained by: file dark_matter_profiles.structure.concentration.WDM.F90
Modules used: transfer_functions
function: wdmdefaultconstructor
  Description: Default constructor for the wdm dark matter halo profile concentration class.
  Code lines: 30
  Contained by: file dark_matter_profiles.structure.concentration.WDM.F90
  Modules used: input_parameters

function: wdmgenericconstructor
  Description: Generic constructor for the WDM dark matter halo concentration class.
  Code lines: 9
  Contained by: file dark_matter_profiles.structure.concentration.WDM.F90

  Description: An implementation of dark matter halo profile concentrations using the Zhao et al. [2009] algorithm.
  Code lines: 103

interface: darkmatterprofileconcentrationzhao2009
  Description: Constructors for the zhao2009 dark matter halo profile concentration class.
  Code lines: 3

function: zhao2009concentration
  Description: Return the concentration of the dark matter halo profile of node using the Zhao et al. [2009] algorithm.
  Code lines: 19
  Modules used: dark_matter_halo_formation_times

function: zhao2009darkmatterprofiledefinition
  Description: Return a dark matter density profile object defining that used in the definition of concentration in the Zhao et al. [2009] algorithm.
  Code lines: 19
  Modules used: dark_matter_halo_scales

function: zhao2009defaultconstructor
  Description: Default constructor for the zhao2009 dark matter halo profile concentration class.
  Code lines: 6
  Modules used: input_parameters

function: zhao2009densitycontrastdefinition
  Description: Return a virial density contrast object defining that used in the definition of concentration in the Zhao et al. [2009] algorithm.
  Code lines: 12

file: dark_matter_profiles.structure.scale.F90
19.1. Program units

**Description:** Contains a module which implements calculations of dark matter profile scale radii from concentrations.

**Code lines:** 178

**module:** dark_matter_profile_scales

**Description:** Implements calculations of dark matter profile scale radii from concentrations.

**Code lines:** 158

**Contained by:** file dark_matter_profiles.structure.scale.F90

**Used by:** subroutine halo_mass_function_compute  subroutine assign_scale_radii

subroutine node_component_dark_matter_- profile_scale_initialize_scale

**function:** dark_matter_profile_scale

**Description:** Compute the scale radius of the dark matter profile of node.

**Code lines:** 143

**Contained by:** module dark_matter_profile_scales

**Modules used:** cosmology_functions  cosmology_parameters
dark_matter_halo_scales  dark_matter_profiles
dark_matter_profiles_concentration  galacticus_calculations_resets
galacticus_nodes  input_parameters
numerical_constants_math  root_finder
virial_density_contrast

**function:** densitycontrastrootfunction

**Description:** Root function used to find the radius in a dark matter profile which encloses a density contrast equal to the currently specified density contrast.

**Code lines:** 11

**Contained by:** function dark_matter_profile_scale

**function:** massrootfunction

**Description:** Root function used to find the mass of a halo corresponding to the definition used for a particular concentration class.

**Code lines:** 32

**Contained by:** function dark_matter_profile_scale

**file:** dark_matter_profiles.structure.shape.F90

**Description:** Contains a module which implements calculations of dark matter halo density profile shapes.

**Code lines:** 113

**module:** dark_matter_profiles_shapes

**Description:** Implements calculations of dark matter halo density profile shapes.

**Code lines:** 93

**Contained by:** file dark_matter_profiles.structure.shape.F90

**Modules used:** galacticus_nodes  iso_varying_string

**Used by:** subroutine node_component_dark_matter_- profile_scale_shape_initialize_shape

**function:** dark_matter_profile_shape

**Description:** Returns the shape of the dark matter profile of thisNode.
subroutine: dark_matter_shapes_initialize
Description: Initialize the dark matter profile module.
Code lines: 52
Contained by: module dark_matter_profiles_shapes
Modules used: dark_matter_profiles_shapes_gao2008 galacticus_error

Description: Contains a module which implements the Gao et al. [2008] Einasto halo shape algorithm.
Code lines: 71

module: dark_matter_profiles_shapes_gao2008
Description: Implements the Gao et al. [2008] Einasto halo shape algorithm.
Code lines: 51
Used by: subroutine dark_matter_shapes_gao2008_initialize

function: dark_matter_profile_shape_gao2008
Description: Returns the Einasto shape parameter, \( \alpha \), of the dark matter profile of thisNode using the method of Gao et al. [2008]. More specifically, the parameter is given by:

\[
\alpha = \begin{cases} 
0.155 + 0.0095\nu^2 & \text{if } \nu < 3.907 \\
0.3 & \text{if } \nu \geq 3.907 
\end{cases} 
\]

(19.11)

where \( \nu = \delta_c(t)/\sigma(M) \) is the peak height of the halo.

Code lines: 26
Contained by: module dark_matter_profiles_shapes_gao2008
Modules used: critical_overdensity galacticus_nodes power_spectra

subroutine: dark_matter_shapes_gao2008_initialize
Description: Initializes the “Gao2008” halo shape module.
Code lines: 10
Contained by: module dark_matter_profiles_shapes_gao2008
Modules used: iso_varying_string

file: dark_matter_profiles.structure_tasks.F90
Description: Contains a module which implements structure tasks related to the dark matter halo density profile.
Code lines: 215

module: dark_matter_profile_structure_tasks
Description: Implements structure tasks related to the dark matter halo density profile.
Code lines: 195
Contained by: file dark_matter_profiles.structure_tasks.F90
19.1. Program units

Modules used: dark_matter_profiles galacticus_nodes
Used by: function galactic_structure_density function galactic_structure_enclosed_mass
function galactic_structure_potential function galactic_structure_rotation_curve
function galactic_structure_rotation_curve_gradient

function: dark_matter_profile_density_task
Description: Computes the density at a given position for a dark matter profile.
Code lines: 44
Contained by: module dark_matter_profile_structure_tasks
Modules used: cosmology_parameters galactic_structure_initial_radii
galactic_structure_options

function: dark_matter_profile_enclosed_mass_task
Description: Computes the mass within a given radius for a dark matter profile.
Code lines: 51
Contained by: module dark_matter_profile_structure_tasks
Modules used: cosmology_parameters galactic_structure_initial_radii
galactic_structure_options

function: dark_matter_profile_potential_task
Description: Return the potential due to dark matter.
Code lines: 22
Contained by: module dark_matter_profile_structure_tasks
Modules used: galactic_structure_options galacticus_error

class: galacticus_nodes
Description: Contains a module which handles node branch jump events.
Code lines: 54
Contained by: file events.branch_jump.F90

module: dark_matter_profile_structure_tasks
Description: Handles satellite node branch jump events.
Code lines: 20
Contained by: file events.branch_jump.F90
Used by: subroutine create_branch_jump_event

**function: node_branch_jump**
Description: Moves a satellite node to a different branch of the merger tree.
Code lines: 44
Contained by: module node_branch_jumps
Modules used: galacticus_display
iso_varying_string
string_handling
galacticus_nodes
merger_trees_evolve_deadlock_status

file: events.halo_formation.F90
Description: Contains a module which performs tasks associated with “halo formation” events.
Code lines: 68

**module: events_halo_formation**
Description: Performs tasks associated with “halo formation” events.
Code lines: 48
Contained by: file events.halo_formation.F90
Used by: subroutine node_componentformation_time_cole2000_create

**subroutine: event_halo_formation**
Description: Perform tasks associated with a “halo formation” event in thisNode.
Code lines: 38
Contained by: module events_halo_formation
Modules used: galacticus_nodes
node_component_hot_halo_cold_mode
node_component_hot_halo_standard
node_component_satellite_standard
node_component_satellite_very_simple

file: events.node_merger.single_level_hierarchy.F90
Description: Contains a module which implements merger of nodes utilizing a single level substructure hierarchy.
Code lines: 99

**module: events_node_mergers_slh**
Description: Implements merger of nodes utilizing a single level substructure hierarchy.
Code lines: 79
Contained by: file events.node_merger.single_level_hierarchy.F90
Used by: subroutine events_node_merger

**subroutine: events_node_merger_do_slh**
Description: Processes a node merging event, utilizing a single level substructure hierarchy.
Code lines: 54
Contained by: module events_node_mergers_slh
Modules used: galacticus_error
galacticus_nodes
iso_varying_string
string_handling
satellite_promotion

**subroutine: events_node_merger_initialize_slh**
19.1. Program units

**Description:** Determine if use of this method is requested and set procedure pointer appropriately if it is.

**Code lines:** 10

**Contained by:** module events_node_mergers_slh

**Modules used:** iso_varying_string

**file:** events.node_promotion.index_shift.F90

**Description:** Contains a module which optionally shifts the index of a node about to be promoted to its parent node, allowing indices to be tracked along merger trees.

**Code lines:** 80

**module:** node_promotion_index_shifts

**Description:** Implements optional shifting of the index of a node about to be promoted to its parent node, allowing indices to be tracked along merger trees.

**Code lines:** 59

**Contained by:** file events.node_promotion.index_shift.F90

**Used by:** subroutine tree_node_promote

**subroutine:** node_promotion_index_shift

**Description:** Shifts the index of thisNode to its parent node just prior to promotion, thereby allowing indices to track galaxies through the tree.

**Code lines:** 39

**Contained by:** module node_promotion_index_shifts

**Modules used:** galacticus_nodes input_parameters

**file:** events.subhalo_promotion.F90

**Description:** Contains a module which handles node subhalo promotion events.

**Code lines:** 71

**module:** node_subhalo_promotions

**Description:** Handles subhalo promotion events.

**Code lines:** 51

**Contained by:** file events.subhalo_promotion.F90

**Used by:** subroutine scan_for_subhalo_promotions

**function:** node_subhalo_promotion

**Description:** Promotes a subhalo to be an isolated node.

**Code lines:** 41

**Contained by:** module node_subhalo_promotions

**Modules used:** galacticus_display galacticus_nodes iso_varying_string merger_trees_evolve_deadlock_status merger_trees_evolve_node string_handling

**file:** galactic_dynamics.bar_instability.ELN.F90

**Description:** Contains a module which implements calculations of bar instability based on the Efstathiou et al. [1982] criterion.

**Code lines:** 128

**module:** galactic_dynamics_bar_instabilities_eln

**Description:** Implements calculations of bar instability based on the Efstathiou et al. [1982] criterion.
19. Source Code Documentation

**Code lines:** 108
**Contained by:** file `galactic_dynamics.bar_instability.ELN.F90`
**Used by:** subroutine `galactic_dynamics_bar_instability_initialize`

**subroutine: bar_instability_timescale_eln**
**Description:** Computes a timescale for depletion of a disk to a pseudo-bulge via bar instability based on the criterion of Efstathiou et al. [1982].
**Code lines:** 54
**Contained by:** module `galactic_dynamics_bar_instabilities_eln`
**Modules used:** `galacticus_nodes`
**Modules used:** `numerical_constants_astronomical`
**Modules used:** `numerical_constants_physical`

**subroutine: galactic_dynamics_bar_instabilities_eln_initialize**
**Description:** Initializes the “ELN” bar instability module.
**Code lines:** 36
**Contained by:** module `galactic_dynamics_bar_instabilities_eln`
**Modules used:** `input_parameters`
**Modules used:** `iso_varying_string`

**file: galactic_dynamics.bar_instability.ELN_tidal.F90**
**Description:** Contains a module which implements calculations of bar instability based on the Efstathiou et al. [1982] criterion, but including the effects of tidal forces.
**Code lines:** 158

**module: galactic_dynamics_bar_instabilities_eln_tidal**
**Description:** Implements calculations of bar instability based on the Efstathiou et al. [1982] criterion, but including the effects of tidal forces.
**Code lines:** 137
**Contained by:** file `galactic_dynamics.bar_instability.ELN_tidal.F90`
**Modules used:** `galacticus_nodes`
**Used by:** subroutine `galactic_dynamics_bar_instability_initialize`

**subroutine: bar_instability_timescale_eln_tidal**
**Description:** Computes a timescale for depletion of a disk to a pseudo-bulge via bar instability based on the criterion of Efstathiou et al. [1982], but including an additional term due to external tidal forces.
**Code lines:** 67
**Contained by:** module `galactic_dynamics_bar_instabilities_eln_tidal`
**Modules used:** `galacticus_nodes`
**Modules used:** `numerical_constants_astronomical`
**Modules used:** `numerical_constants_physical`
**Modules used:** `satellites_tidal_fields`

**subroutine: galactic_dynamics_bar_instabilities_eln_tidal_initialize**
**Description:** Initializes the “ELN+tidal” bar instability module.
**Code lines:** 47
**Contained by:** module `galactic_dynamics_bar_instabilities_eln_tidal`
**Modules used:** `input_parameters`
**Modules used:** `iso_varying_string`

**file: galactic_dynamics.bar_instability.F90**
19.1. Program units

**Description:** Contains a module which implements calculations of bar instability in galactic disks.

**Code lines:** 111

**module: galactic_dynamics_bar_instabilities**

**Description:** Implements calculations of bar instability in galactic disks.

**Code lines:** 91

**Contained by:** file galactic_dynamics.bar_instability.F90

**Modules used:** galacticus_nodes iso_varying_string

**Used by:** subroutine node_component_disk_,-- exponential_rate_compute subroutine node_component_dynamics_,-- statistics_bars_record

**subroutine: bar_instability_timescale**

**Description:** Returns a timescale on which the bar instability depletes material from a disk into a pseudo-bulge. A negative value indicates no instability. Also returns the net torque due to any external force causing this instability.

**Code lines:** 12

**Contained by:** module galactic_dynamics_bar_instabilities

**subroutine: galactic_dynamics_bar_instability_initialize**

**Description:** Initialize the bar instability module.

**Code lines:** 56

**Contained by:** module galactic_dynamics_bar_instabilities

**Modules used:** galactic_dynamics_bar_instabilities_,-- eln galactic_dynamics_bar_instabilities_,-- eln_tidal galacticus_error null input_parameters

**file: galactic_dynamics.bar_instability.null.F90**

**Description:** Contains a module which implements a null calculation of bar instability.

**Code lines:** 57

**module: galactic_dynamics_bar_instabilities_null**

**Description:** Implements a null calculation of bar instability.

**Code lines:** 37

**Contained by:** file galactic_dynamics.bar_instability.null.F90

**Modules used:** galacticus_nodes

**Used by:** subroutine galactic_dynamics_bar_,-- instability_initialize

**subroutine: bar_instability_timescale_null**

**Description:** Assumes that disks are never bar unstable and so returns an infinite timescale for bar instability.

**Code lines:** 11

**Contained by:** module galactic_dynamics_bar_instabilities_null

**subroutine: galactic_dynamics_bar_instabilities_null_initialize**

**Description:** Initializes the “Null” bar instability module.

**Code lines:** 10
19. Source Code Documentation

**Contained by:** module galactic_dynamics_bar_instabilities_null

**Modules used:** iso_varying_string

**file:** galactic_structure.density.F90

**Description:** Contains a module which implements calculations of the density at a specific position.

**Code lines:** 142

**module:** galactic_structure_densities

**Description:** Implements calculations of the density at a specific position.

**Code lines:** 122

**Contained by:** file galactic_structure.density.F90

**Used by:**

- function black_hole_binary_separation_growth_rate_standard
- function velocity_dispersion_integrand
- function galacticus_output_trees_vlcty_vlcty_density_inTEGRAND
- function galacticus_output_trees_vlcty_density_inTEGRAND1
- subroutine node_component_black_hole_standard_mass_accretion_rate
- subroutine node_component_hot_halo_cold_mode_rate_compute
- function ram_pressure_stripping_mass_loss_rate_spheroid_simple
- function satellites_tidal_fields_spherical.symmetry.get
- function satellite_tidal_stripping_rate_zentner2005

**function:** component_density

**Description:** Unary function returning the density in a component. Suitable for mapping over components.

**Code lines:** 8

**Contained by:** module galactic_structure_densities

**Modules used:** galacticus_nodes

**function:** galactic_structure_density

**Description:** Compute the density (of given massType) at the specified position. Assumes that galactic structure has already been computed.

**Code lines:** 97

**Contained by:** module galactic_structure_densities

**Modules used:**

- coordinate_systems
- dark_matter_profile_structure_tasks
- galactic_structure_options
- galacticus_error
- galacticus_nodes
- hot_halo_mass_distributions
- node_component_hot_halo_cold_mode_structure_tasks

**file:** galactic_structure.enclosed_mass.F90
Description: Contains a module which implements calculations of the mass enclosed within a specified radius.
Code lines: 211

module: galactic_structure_enclosed_masses
Description: Implements calculations of the mass enclosed within a specified radius.
Code lines: 191
Contained by: file galactic_structure.enclosed_mass.F90
Modules used: galactic_structure_options galacticus_nodes
Used by: function black_hole_binary_initial_radius tidal_radius_root
        function galactic_structure_velocity_dispersion
        subroutine galacticus_output_analysis_mass_dpndnt_sz_dstrbtins
        subroutine galacticus_output_tree_density_contrast
        subroutine galacticus_output_tree_output_filter_luminosity
        subroutine galacticus_output_tree_output_filter_stellar_mass
        subroutine galacticus_output_tree_half_light
        subroutine galacticus_output_tree_mass_profile
        subroutine galacticus_output_tree_rotation_curve
        subroutine galacticus_output_tree_velocity_dispersion
        function hot_halo_ram_pressure_stripping_radius_solver
        subroutine merger_tree_history_store
        subroutine node_component_hot_halo_cold_mode_node_merger
        subroutine node_component_satellite_orbiting_rate_compute
        subroutine node_component_satellite_orbiting_scale_set
        subroutine satellite_merging_mass_movement_baugh2005
        subroutine satellite_merging_remnant_size_cole2000
        subroutine satellite_merging_remnant_progenitor_properties_cole2000
        function satellites_tidal_fields_spherical_symmetry_get
        function satellite_tidal_stripping_rate_zentner2005
        function component_enclosed_mass
Description: Unary function returning the enclosed mass in a component. Suitable for mapping over components.
Code lines: 7
Contained by: module galactic_structure_enclosed_masses
function: enclosed_mass_root
Description: Root function used in solving for the radius that encloses a given mass.
Code lines: 6
Contained by: module galactic_structure_enclosed_masses

function: galactic_structure_enclosed_mass
Description: Solve for the mass within a given radius, or the total mass if no radius is specified. Assumes that galactic structure has already been computed.
Code lines: 55
Contained by: module galactic_structure_enclosed_masses
Modules used: dark_matter_profile_structure_tasks hot_halo_mass_distributions
node_component_hot_halo_cold_mode_
structure_tasks

subroutine: galactic_structure_enclosed_mass_defaults
Description: Set the default values for options in the enclosed mass functions.
Code lines: 37
Contained by: module galactic_structure_enclosed_masses
Modules used: galacticus_error

function: galactic_structure_radius_enclosing_mass
Description: Return the radius enclosing a given mass (or fractional mass) in thisNode.
Code lines: 60
Contained by: module galactic_structure_enclosed_masses
Modules used: dark_matter_halo_scales galacticus_display
galacticus_error iso_varying_string
root_finder string_handling

file: galactic_structure.options.F90
Description: Contains a module which provides various internal option codes for the galactic structure functions.
Code lines: 136

module: galactic_structure_options
Description: Provides various internal option codes for the galactic structure functions.
Code lines: 116
Contained by: file galactic_structure.options.F90
Used by: function black_hole_binary_initial_-radius_tidal_radius
function black_hole_binary_separation_-growth_rate_standard
function dark_matter_profile_density_-task
function dark_matter_profile_enclosed_-mass_task
function dark_matter_profile_rotation_-curve_gradient_task
function dark_matter_profile_rotation_-curve_task
function galactic_structure_density module galactic_structure_enclosed_-masses
module galactic_structure_potentials
module galactic_structure_initial_radii_adiabatic
module galactic_structure_rotation_curve_gradients
function galactic_structure_velocity_dispersion
module galacticus_output_analyses_mass_distributions
subroutine galacticus_output_tree_density_contrast
subroutine galacticus_merger_tree_output_filter_luminosity
subroutine galacticus_merger_tree_output_filter_stellar_mass
subroutine galacticus_output_tree_half_light
subroutine galacticus_output_tree_mass_profile
subroutine galacticus_output_tree_rotation_curve
subroutine galacticus_output_tree_rotation_curve_initialize
subroutine galacticus_output_tree_velocity_dispersion
subroutine galacticus_output_tree_velocity_dispersion_initialize
function galacticus_output_trees_velocity_dispersion_lambdar_intgrnd1
function galacticus_output_trees_velocity_dispersion_lambdar_intgrnd2
function hot_halo_mass_distribution_density_task
function hot_halo_mass_distribution_rotation_curve_task
function hot_halo_ram_pressure_stripping_radius_solver
subroutine merger_tree_history_store
subroutine solve_for_radius
module galactic_structure_rotation_curves
module galactic_structure_surface_densities
module galacticus_output_analyses_mass_distributions
subroutine galacticus_output_tree_density_contrast
function mean_density_contrast_root
subroutine galacticus_merger_tree_output_filter_stellar_mass
subroutine galacticus_output_tree_half_mass
subroutine galacticus_output_tree_rotation_mass
subroutine galacticus_output_tree_rotation_curve
subroutine galacticus_output_tree_rotation_curve_initialize
subroutine galacticus_output_tree_velocity_dispersion
function galacticus_output_trees_velocity_dispersion_lambdar_intgrnd
function galacticus_output_trees_velocity_dispersion_lambdar_intgrnd2
function hot_halo_mass_distribution_enclosed_mass_task
function hot_halo_mass_distribution_rotation_curve_task
function intergalactic_medium_state_internal_update
subroutine merger_tree_record_evolution_store
function node_component_black_hole_simple_enclosed_mass
function node_component_black_hole_standard_enclosed_mass
function node_component_disk_exponential_density
function node_component_disk_exponential_enclosed_mass
function node_component_disk_exponential_rotation_curve
function node_component_disk_exponential_rotation_curve_gradient
function node_component_disk_standard_enclosed_mass
function node_component_disk_standard_density
function node_component_disk_standard_potential
function node_component_disk_standard_rotation_curve
function node_component_disk_varying_density
function node_component_disk_varying_enclosed_mass
function node_component_disk_varying_rotation_curve
function node_component_disk_varying_rotation_curve_gradient
function node_component_spheroid_standard_density
function node_component_spheroid_standard_enclosed_mass
function node_component_spheroid_standard_rotation_curve
function node_component_spheroid_standard_potential
function node_component_spheroid_standard_rotation_curve_gradient
19. Source Code Documentation

function node_component_black_hole_-simple_potential
function node_component_black_hole_-simple_rotation_curve_gradient
subroutine node_component_black_hole_-standard_merge_black_holes
subroutine node_component_black_hole_-standard_triple_interaction
function node_component_black_hole_-standard_rotation_curve
subroutine node_component_disk_-exponential_rate_compute
subroutine node_component_hot_halo_-cold_mode_outflow_return
function node_component_hot_halo_-cold_mode_density_task
function node_component_hot_halo_-cold_mode_rotation_curve_task
subroutine node_component_hot_halo_-cold_mode_rate_compute
subroutine node_component_hot_halo_-cold_mode_node_merger
subroutine node_component_hot_halo_-cold_mode_outflow_return
function node_component_hot_halo_-cold_mode_rotation_curve_task
subroutine node_component_hot_halo_-cold_mode_rate_compute
subroutine node_component_hot_halo_-cold_mode_enclosed_mass_task
function node_component_hot_halo_-cold_mode_rotation_curve_task
subroutine node_component_satellite_-orbiting_rate_compute
function radiation_intergalactic_-background_internal_update
function ram_pressure_stripping_mass_-loss_rate_disk_simple
function satellite_dynamical_friction_-acceleration_chandrasekhar
subroutine satellite_merging_mass_-movement_baugh2005
subroutine satellite_merging_mass_-movement_simple
function half_mass_radius_root_cole2000
subroutine satellite_merging_remnant_-progenitor_properties_cole2000
function satellite_tidal_fields_-spherical_symmetry_get
function imf_select_disk_spheroid
function star_formation_rate_surface_-density_disk_br
function star_formation_rate_surface_-density_disk_ks
function testfuncdouble0
function tidal_stripping_mass_loss_-rate_spheroid_simple
function node_component_black_hole_-simple_rotation_curve
subroutine node_component_black_hole_-standard_mass_accretion_rate
subroutine node_component_black_hole_-standard_satellite_merging
function node_component_black_hole_-standard_potential
function node_component_black_hole_-standard_rotation_curve_gradient
subroutine node_component_hot_halo_-cold_mode_node_merger
subroutine node_component_hot_halo_-cold_mode_rate_compute
function node_component_hot_halo_-cold_mode_enclosed_mass_task
function node_component_hot_halo_-cold_mode_rotation_curve_task
subroutine node_component_satellite_-orbiting_rate_compute
function radiation_intergalactic_-background_internal_update
function ram_pressure_stripping_mass_-loss_rate_spheroid_simple
subroutine satellite_merging_mass_-movement_baugh2005
subroutine satellite_merging_remnant_-size_cole2000
subroutine satellite_merging_remnant_-progenitor_properties_standard
function satellites_tidal_fields_-spherical_symmetry_get
function satellite_tidal_heating_rate_-gnedin
function star_formation_feedback_disk_-outflow_rate_creevey2012_integrand
function star_formation_rate_surface_-density_disk_kmt09
function star_formation_rate_surface_-density_disk_exschmidt
function tidal_stripping_mass_loss_-rate_disk_simple
function: galactic_structure_component_type_decode
Description: Decode a component type from a string, returning the appropriate identifier.
Code lines: 16
Contained by: module galactic_structure_options
Modules used: galacticus_error

function: galactic_structure_mass_type_decode
Description: Decode a mass type from a string, returning the appropriate identifier.
Code lines: 16
Contained by: module galactic_structure_options
Modules used: galacticus_error

file: galactic_structure.potential.F90
Description: Contains a module which implements calculations of gravitational potential.
Code lines: 127

module: galactic_structure_potentials
Description: Implements calculations of the gravitational potential.
Code lines: 93
Contained by: file galactic_structure.potential.F90
Modules used: galactic_structure_options galacticus_nodes
Used by: subroutine node_component_black_hole_standard_merge_black_holes subroutine node_component_black_hole_standard_satellite_merging subroutine node_component_black_hole_standard_triple_interaction function satellite_orbit_convert_to_current_potential

function: component_potential
Description: Unary function returning the potential in a component. Suitable for mapping over components.
Code lines: 7
Contained by: module galactic_structure_potentials

function: galactic_structure_potential
Description: Solve for the gravitational potential at a given radius. Assumes the galactic structure has already been computed.
Code lines: 67
Contained by: module galactic_structure_potentials
Modules used: dark_matter_profile_structure_tasks node_component_black_hole_simple_structure node_component_black_hole_standard_structure_tasks

file: galactic_structure.radius_solver.F90
Description: Contains a module which implements calculations of sizes of galactic components (or more general components).
Code lines: 113

module: galactic_structure_radii
19. Source Code Documentation

**Description:** Implements calculations of sizes of galactic components (or more general components).

**Code lines:** 93

**Contained by:** file `galactic_structure.radius_solver.F90`

**Modules used:** `galacticus_nodes` `iso_varying_string`

**Used by:** subroutine `galacticus_merger_tree-output`
subroutine `tree_node_compute-derivatives`
subroutine `satellite_merging_remnant-progenitor_properties_cole2000`
subroutine `satellite_merging_remnant-progenitor_properties_standard`

**subroutine:** `galactic_structure_radii_solve`

**Description:** Solve for the radii of galactic components in **thisNode**.

**Code lines:** 63

**Contained by:** module `galactic_structure_radii`

**Modules used:** `galactic_structure_radii_adiabatic` `galactic_structure_radii_fixed`
`galactic_structure_radii_linear` `galactic_structure_radii_simple`
`galacticus_error` `input_parameters`

**file:** `galactic_structure.radius_solver.adiabatic.F90`

**Description:** Contains a module which implements an adiabatic contraction galactic radii solver (including self-gravity of baryons) using the adiabatic contraction algorithm of Gnedin et al. [2004].

**Code lines:** 352

**module:** `galactic_structure_radii_adiabatic`

**Description:** Implements an adiabatic contraction galactic radii solver (including self-gravity of baryons) using an adiabatic contraction algorithm.

**Code lines:** 331

**Contained by:** file `galactic_structure.radius_solver.adiabatic.F90`

**Modules used:** `galactic_structure_radius_solver-procedures`

**Used by:** subroutine `galactic_structure_radii-solve`

**subroutine:** `galactic_structure_radii_adiabatic_initialize`

**Description:** Initializes the “adiabatic” galactic radii solver module.

**Code lines:** 46

**Contained by:** module `galactic_structure_radii_adiabatic`

**Modules used:** `input_parameters` `iso_varying_string`

**subroutine:** `galactic_structure_radii_solve_adiabatic`

**Description:** Find the radii of galactic components in **thisNode** using the “adiabatic” method.

**Code lines:** 105

**Contained by:** module `galactic_structure_radii_adiabatic`

**Modules used:** `cosmology_parameters` `galacticus_display`
`galacticus_error` `node_component_dark_matter_profile-scale`
`node_component_disk_exponential` `node_component_spheroid_standard`

**subroutine:** `solve_for_radius`

**Description:** Solve for the equilibrium radius of the given component.
19.1. Program units

**Code lines:** 150  
**Contained by:** module `galactic_structure_radiiadiabatic`  
**Modules used:**  
- `dark_matter_profiles`  
- `galactic_structure_options`  
- `galacticus_error`  
- `memory_management`  
- `numerical_constants_physical`  
- `string_handling`  

**file:** `galactic_structure.radius_solver.fixed.F90`  
**Description:** Contains a module which implements a “fixed” galactic radii solver in which sizes are always equal to the halo virial radius multiplied by its spin parameter and a multiplicative constant.  
**Code lines:** 151

**module:** `galactic_structure_radii_fixed`  
**Description:** Implements a “fixed” galactic radii solver in which sizes are always equal to the halo virial radius multiplied by its spin parameter and a multiplicative constant.  
**Code lines:** 130  
**Contained by:** file `galactic_structure.radius_solver.fixed.F90`  
**Modules used:** `galactic_structure_radius_solver_procedures`  
**Used by:** subroutine `galactic_structure_radii_solve`  

**subroutine:** `galactic_structure_radii_fixed_initialize`  
**Description:** Initializes the “fixed” galactic radii solver module.  
**Code lines:** 23  
**Contained by:** module `galactic_structure_radii_fixed`  
**Modules used:** `input_parameters`  

**subroutine:** `galactic_structure_radii_solve_fixed`  
**Description:** Find the radii of galactic components in `thisNode` using the “fixed” method.  
**Code lines:** 59  
**Contained by:** module `galactic_structure_radii_fixed`  
**Modules used:**  
- `node_component_dark_matter_profile_scale`  
- `node_component_disk_exponential`  
- `node_component_spheroid_standard`  

**subroutine:** `solve_for_radius`  
**Description:** Solve for the equilibrium radius of the given component.  
**Code lines:** 26  
**Contained by:** module `galactic_structure_radii_fixed`  
**Modules used:** `dark_matter_halo_scales`  

**file:** `galactic_structure.radius_solver.initial_radii.F90`  
**Description:** Contains a module which implements calculations of the initial radius in the dark matter halo for use when solving for galactic structure.  
**Code lines:** 125

**module:** `galactic_structure_initial_radii`
Description: Implements calculations of the initial radius in the dark matter halo for use when solving for galactic structure.

Code lines: 104

Contained by: file `galactic_structure.radius_solver.initial_radii.F90`

Modules used: `galacticus_nodes iso_varying_string`

Used by: function `dark_matter_profile_density_task`

function `dark_matter_profile_enclosed_mass_task`

subroutine `solve_for_radius`

**function: `galactic_structure_radius_initial`**

Description: Find the initial radius in the dark matter halo of `thisNode` corresponding to the given final radius.

Code lines: 11

Contained by: module `galactic_structure_initial_radii`

**function: `galactic_structure_radius_initial_derivative`**

Description: Find the derivative of the initial radius in the dark matter halo of `thisNode` with respect to the final radius corresponding to the given final radius.

Code lines: 12

Contained by: module `galactic_structure_initial_radii`

**subroutine: `galactic_structure_radius_initialInitialize`**

Description: Initialize the initial radius module for the galactic structure subsystem.

Code lines: 55

Contained by: module `galactic_structure_initial_radii`

Modules used: `galactic_structure_initial_radii_adiabatic galactic_structure_initial_radii_static galacticus_error input_parameters`

**file: `galactic_structure.radius_solver.initial_radii.adiabatic.F90`**

Description: Contains a module which implements calculations of initial radius in the dark matter halo using the adiabatic contraction algorithm of Gnedin et al. [2004].

Code lines: 433

**module: `galactic_structure_initial_radii_adiabatic`**

Description: Implements calculations of initial radius in the dark matter halo using the adiabatic contraction algorithm of Gnedin et al. [2004].

Code lines: 412

Contained by: file `galactic_structure.radius_solver.initial_radii.adiabatic.F90`

Modules used: `galactic_structure_options galacticus_nodes`

Used by: subroutine `galactic_structure_radius_initial_initialize`

**function: `adiabatic_solver_mean_orbital_radius`**

Description: Returns the orbit averaged radius for dark matter corresponding the given radius using the model of Gnedin et al. [2004].

Code lines: 8

Contained by: module `galactic_structure_initial_radii_adiabatic`

**function: `adiabatic_solver_mean_orbital_radius_derivative`**
19.1. Program units

Description: Returns the derivative of the orbit averaged radius for dark matter corresponding to the given radius using the model of Gnedin et al. [2004].

Code lines: 8

Contained by: module galactic_structure_initial_radii_adiabatic

function: component_enclosed_mass

Description: Unary function returning the enclosed mass in a component. Suitable for mapping over components. Ignores the dark matter profile.

Code lines: 13

Contained by: module galactic_structure_initial_radii_adiabatic

function: component_rotation_curve

Description: Unary function returning the squared rotation curve in a component. Suitable for mapping over components.

Code lines: 12

Contained by: module galactic_structure_initial_radii_adiabatic

function: component_rotation_curve_gradient

Description: Unary function returning the squared rotation curve gradient in a component. Suitable for mapping over components.

Code lines: 12

Contained by: module galactic_structure_initial_radii_adiabatic

subroutine: galactic_structure_initial_radii_adiabatic_initialize

Description: Initializes the “adiabatic” initial radii module.

Code lines: 37

Contained by: module galactic_structure_initial_radii_adiabatic

Modules used: input_parameters iso_varying_string

subroutine: galactic_structure_initial_radii_adiabatic_compute_factors

Description: Compute various factors needed when solving for the initial radius in the dark matter halo using the adiabatic contraction algorithm of Gnedin et al. [2004].

Code lines: 177

Contained by: module galactic_structure_initial_radii_adiabatic

Modules used: cosmology_parameters dark_matter_halo_scales
hot_halo_mass_distributions node_component_black_hole_simple_structure
node_component_black_hole_standard_structure_tasks
numerical_constants_physical node_component_hot_halo_cold_mode_structure_tasks

function: galactic_structure_radius_initial_adiabatic

Description: Compute the initial radius in the dark matter halo using the adiabatic contraction algorithm of Gnedin et al. [2004].

Code lines: 31

Contained by: module galactic_structure_initial_radii_adiabatic

Modules used: root_finder

function: galactic_structure_radius_initial_adiabatic_solver
19. Source Code Documentation

Description: Root function used in finding the initial radius in the dark matter halo when solving for adiabatic contraction.
Code lines: 17
Contained by: module galactic_structure_initial_radii_adiabatic
Modules used: dark_matter_profiles

function: galactic_structure_radius_initial_derivative_adiabatic
Description: Compute the derivative of the initial radius in the dark matter halo using the adiabatic contraction algorithm of Gnedin et al. [2004].
Code lines: 31
Contained by: module galactic_structure_initial_radii_adiabatic
Modules used: root_finder

function: galactic_structure_radius_initial_derivative_adiabatic_solver
Description: Root function used in finding the derivative of the initial radius in the dark matter halo when solving for adiabatic contraction.
Code lines: 21
Contained by: module galactic_structure_initial_radii_adiabatic
Modules used: dark_matter_profiles numerical_constants_math

file: galactic_structure.radius_solver.initial_radii.static.F90
Description: Contains a module which implements calculations of initial radius in which the dark matter halo is assumed to be static.
Code lines: 67

module: galactic_structure_initial_radii_static
Description: Implements calculations of initial radius in which the dark matter halo is assumed to be static.
Code lines: 47
Contained by: file galactic_structure.radius_solver.initial_radii.static.F90
Modules used: galacticus_nodes
Used by: subroutine galactic_structure_radius_initial_initialize

subroutine: galactic_structure_initial_radii_static_initialize
Description: Initiates the “static” initial radii module.
Code lines: 13
Contained by: module galactic_structure_initial_radii_static
Modules used: iso_varying_string

function: galactic_structure_radius_initial_derivative_static
Description: Compute the derivative of the initial radius in the dark matter halo assuming the halo is static.
Code lines: 8
Contained by: module galactic_structure_initial_radii_static

function: galactic_structure_radius_initial_static
Description: Compute the initial radius in the dark matter halo assuming the halo is static.
Code lines: 8
Contained by: module galactic_structure_initial_radii_static
file: galactic_structure.radius_solver.linear.F90
Description: Contains a module which implements a “linear” galactic radii solver (no adiabatic contraction and no self-gravity of baryons, and size simply scales in proportion to specific angular momentum).
Code lines: 129

module: galactic_structure_radii_linear
Description: Implements a “linear” galactic radii solver (no adiabatic contraction and no self-gravity of baryons, and size simply scales in proportion to specific angular momentum).
Code lines: 108
Contained by: file galactic_structure.radius_solver.linear.F90
Modules used: galactic_structure_radius_solver_procedures
Used by: subroutine galactic_structure_radii_linear_initialize

subroutine: galactic_structure_radii_linear_initialize
Description: Initializes the “linear” galactic radii solver module.
Code lines: 9
Contained by: module galactic_structure_radii_linear
Modules used: iso_varying_string

subroutine: galactic_structure_radii_solve_linear
Description: Find the radii of galactic components in thisNode using the “linear” method.
Code lines: 57
Contained by: module galactic_structure_radii_linear
Modules used: node_component_dark_matter_profile_scale
node_component_disk_exponential
node_component_spheroid_standard

subroutine: solve_for_radius
Description: Solve for the equilibrium radius of the given component.
Code lines: 23
Contained by: module galactic_structure_radii_linear
Modules used: dark_matter_halo_scales

file: galactic_structure.radius_solver.procedures.F90
Description: Contains a module which holds procedure pointers used by the galactic structure radii solver subsystem.
Code lines: 40

module: galactic_structure_radius_solver_procedures
Description: Holds procedure pointers used by the galactic structure radii solver subsystem.
Code lines: 20
Contained by: file galactic_structure.radius_solver.procedures.F90
Modules used: galacticus_nodes
Used by: module galactic_structure_radii_adiabatic
module galactic_structure_radii_fixed
module galactic_structure_radii_linear module galactic_structure_radii_simple

file: galactic_structure.radius_solver.simple.F90
Description: Contains a module which implements a simple galactic radii solver (no adiabatic contraction and no self-gravity of baryons).
Code lines: 172

module: galactic_structure_radii_simple
Description: Implements a simple galactic radii solver (no adiabatic contraction and no self-gravity of baryons).
Code lines: 152
Contained by: file galactic_structure.radius_solver.simple.F90
Modules used: galactic_structure_radius_solver_procedures
Used by: subroutine galactic_structure_radii_solve

subroutine: galactic_structure_radii_simple_initialize
Description: Initializes the “simple” galactic radii solver module.
Code lines: 23
Contained by: module galactic_structure_radii_simple
Modules used: input_parameters iso_varying_string

subroutine: galactic_structure_radii_solve_simple
Description: Find the radii of galactic components in thisNode using the “simple” method.
Code lines: 79
Contained by: module galactic_structure_radii_simple
Modules used: galacticus_error node_component_dark_matter_profile_scale
node_component_disk_exponential node_component_spheroid_standard

subroutine: solve_for_radius
Description: Solve for the equilibrium radius of the given component.
Code lines: 26
Contained by: module galactic_structure_radii_simple
Modules used: dark_matter_profiles

file: galactic_structure.rotation_curve.F90
Description: Contains a module which implements calculations of the rotation curve as a specified radius.
Code lines: 119

module: galactic_structure_rotation_curves
Description: Implements calculations of the rotation curve as a specified radius.
Code lines: 99
Contained by: file galactic_structure.rotation_curve.F90
Modules used: galacticus_structure_options galacticus_nodes
Used by: function black_hole_binary_separation_growth_rate_standard subroutine solve_for_radius growth_rate_standard
function galactic_structure_rotation_curve_gradient
Description: Unary function returning the squared rotation curve in a component. Suitable for mapping over components.
Code lines: 7
Contained by: module galactic_structure_rotation_curves

function galacticus_output_trees_rotation_curve
function galacticus_output_trees_vlcty_dsprsn_lambdar_intgrnd1
function galacticus_output_trees_vlcty_dsprsn_lambdar_intgrnd2
function tidal_stripping_mass_loss_rate_disk_simple
function tidal_stripping_mass_loss_rate_spheroid_simple

function: component_rotation_curve
Description: Solve for the rotation curve a given radius. Assumes that galacticus structure has already been solved for.,
Code lines: 73
Contained by: module galactic_structure_rotation_curves
Modules used: dark_matter_profile_structure_tasks hot_halo_mass_distributions
node_component_black_hole_simple_structure
node_component_hot_halo_cold_mode_structure

file: galactic_structure.rotation_curve.gradient.F90
Description: Contains a module which implements calculations of the gradient of the rotation curve.
Code lines: 135

module: galactic_structure_rotation_curve_gradients
Description: Implements calculations of the rotation curve gradient
Code lines: 101
Contained by: file galactic_structure.rotation_curve.gradient.F90
Modules used: galactic_structure_options galacticus_nodes
Used by: function black_hole_binary_separation_growth_rate_standard

function: component_rotation_curve_gradient
Description: Unary function returning the gradient of the squared rotation curve in a component. Suitable for mapping over components.
Code lines: 7
Contained by: module galactic_structure_rotation_curve_gradients

function: galactic_structure_rotation_curve_gradient
Description: Solve for the rotation curve gradient at a given radius. Assumes the galactic structure has already been computed.
Code lines: 76
Contained by: module galactic_structure_rotation_curve_gradients
Modules used: dark_matter_profile_structure_tasks galactic_structure_rotation_curves
hot_halo_mass_distributions node_component_black_hole_simple_structure
19. Source Code Documentation

```plaintext
node_component_black_hole_standard_structure_tasks
node_component_hot_halo_cold_mode_structure_tasks
```

**file: galactic_structure.surface_density.F90**

**Description:** Contains a module which implements calculations of the surface density at a specific position.

**Code lines:** 130

**module: galactic_structure_surface_densities**

**Description:** Implements calculations of the surface density at a specific position.

**Code lines:** 110

**Contained by:** file `galactic_structure.surface_density.F90`

**Modules used:** `galactic_structure_options` `galacticus_nodes`

**Used by:**
- function `galacticus_output_trees_vlcty_dsprsn_lambdar_intgrnd1`
- function `galacticus_output_trees_vlcty_dsprsn_lambdar_intgrnd2`
- function `ram_pressure_stripping_mass_loss_rate_disk_simple`
- function `star_formation_feedback_disk_density_disk_br`
- function `star_formation_feedback_disk_density_disk_ks`
- function `star_formation_feedback_disk_density_disk_exschmidt`

**function: component_surface_density**

**Description:** Unary function returning the surface density in a component. Suitable for mapping over components.

**Code lines:** 7

**Contained by:** module `galactic_structure_surface_densities`

**function: galactic_structure_surface_density**

**Description:** Compute the density (of given `massType`) at the specified `position`. Assumes that galactic structure has already been computed.

**Code lines:** 84

**Contained by:** module `galactic_structure_surface_densities`

**Modules used:** `coordinate_systems` `galacticus_error`

**file: galactic_structure.velocity_dispersions.F90**

**Description:** Contains a module which implements calculations of the velocity dispersions in isotropic spherical systems by solving the Jeans equation.

**Code lines:** 99

**module: galactic_structure_velocity_dispersions**

**Description:** Implements calculations of the velocity dispersions in isotropic spherical systems by solving the Jeans equation.

**Code lines:** 76

**Contained by:** file `galactic_structure.velocity_dispersions.F90`

**Modules used:** `galacticus_nodes`

**Used by:**
- function `black_hole_binary_separation_growth_rate_standard` subroutine `galacticus_output_tree_velocity_dispersion`
- function `galacticus_output_trees_vlcty_dsprsn_lambdar_intgrnd1`
- function `galacticus_output_trees_vlcty_dsprsn_vlcty_dnsty_srfc_intgrnd`
function: galactic_structure_velocity_dispersion
Description: Returns the velocity dispersion of the specified componentType in thisNode at the given radius.
Code lines: 41
Contained by: module galactic_structure_velocity_dispersions
Modules used: fgs1 galactic_structure_densities
galactic_structure_enclosed_masses galactic_structure_options
iso_c_binding numerical_integration

function: velocity_dispersion_integrand
Description: Integrand function used for finding velocity dispersions using Jeans equation.
Code lines: 17
Contained by: module galactic_structure_velocity_dispersions
Modules used: galactic_structure_densities galactic_structure_enclosed_masses
iso_c_binding numerical_constants_physical

file: galacticus.banner.F90
Description: Contains a module which displays a banner for GALACTICUS.
Code lines: 46

module: galacticus_banner
Description: Displays a banner for GALACTICUS.
Code lines: 26
Contained by: file galacticus.banner.F90
Used by: program galacticus

subroutine: galacticus_banner_show
Description: Displays the GALACTICUS banner.
Code lines: 16
Contained by: module galacticus_banner

file: galacticus.calculations_reset.F90
Description: Contains a module which handles resetting of calculations before a new or updated node is processed.
Code lines: 85

module: galacticus_calculations_resets
Description: Handles resetting of calculations before a new or updated node is processed.
Code lines: 65
Contained by: file galacticus.calculations_reset.F90
Used by: function dark_matter_profile_scale subroutine halo_mass_function_compute
subroutine tree_node_compute_derivatives subroutine tree_node_evolve

subroutine: galacticus_calculations_reset
Description: Calls any routines required to reset all calculation for a new or updated node.
Code lines: 55
Contained by: module galacticus_calculations_resets
19. Source Code Documentation

**Modules used:**
- `cooling_radii_isothermal`
- `cooling_specificAngular_momenta_constant_rotation`
- `dark_matter_profiles`
- `node_component_disk_exponential`
- `node_component_hot_halo_very_simple`
- `star_formation_rate_surface_density_disks_exschmidt`
- `star_formation_rate_surface_density_disks_ks`
- `star_formation_rate_surface_density_disks_br`
- `star_formation_rate_surface_density_disks_kmt09`
- `star_formation_timescale_disks_halo_scaling`
- `cooling_radii_simple`
- `dark_matter_halo_scales`
- `galacticus_nodes`
- `node_component_hot_halo_standard`
- `star_formation_rate_surface_density_disks_br`
- `star_formation_rate_surface_density_disks_kmt09`
- `star_formation_timescale_disks_halo_scaling`

**file:** `galacticus.display.F90`

*Description:* Contains a module which implements outputting of formatted, indented messages at various verbosity levels from GALACTICUS.

*Code lines:* 346

**module:** `galacticus_display`

*Description:* Implements outputting of formatted, indented messages at various verbosity levels from GALACTICUS.

*Code lines:* 326

*Contained by:* file `galacticus.display.F90`

*Modules used:* `iso_varying_string`

*Used by:* program `tests_excursion_sets`

-subroutine `chemical_state_cie_file_read`

-subroutine `cooling_function_cie_file_read`

-subroutine `cooling_function_not_matched`

-function `cosmologyparameterssimpledefaultconstructor`

-subroutine `einastofourierprofiletablemake`

-subroutine `einastofreefalltabulate`

-function `node_subhalo_promotion`

-subroutine `galactic_structure_radii_solve_adiabatic`

-function `galacticus_task_evolve_tree`

-subroutine `geometry_survey_window_functions_caputi_2011_ukidss_uds`

-subroutine `geometry_survey_window_functions_li_white_2009_sdss`

-subroutine `halo_mass_function_compute`

-function `generalized_press_schechter_branch_mass`

-subroutine `merger_tree_build_initialize`

-subroutine `merge_tree_construct_fully_specified`
19.1. Program units

*subroutine* `assign_scale_radii`
*subroutine* `create_node_indices`
*subroutine* `scan_for_branch_jumps`

*subroutine* `build_descendent_pointers`
*subroutine* `merger_tree_read_initialize`
*subroutine* `time_until_merging__subresolution`

*subroutine* `galacticusimport`
*subroutine* `galacticusopen`
*subroutine* `sussingopen`

*function* `evolve_to_time`
*subroutine* `events_node_merger`
*subroutine* `tree_node_promote`

*subroutine* `satellite_merger_process`
*function* `root_finder_find`

*subroutine* `galacticusimport`
*subroutine* `galacticusopen`
*subroutine* `sussingtreeindicesread`
*subroutine* `sussingnodeindicesread`
*subroutine* `galacticusopen`
*subroutine* `merger_tree_evolve_to`
*subroutine* `tree_node_odes_error_handler`
*subroutine* `evolve_to_time_report`

*subroutine* `galacticusimport`
*subroutine* `galacticusopen`
*subroutine* `sussingtreeindicesread`

*subroutine* `sussingopen`
*subroutine* `sussingtreeindicesread`
*subroutine* `galacticusopen`
*subroutine* `merger_tree_evolve_to`

*subroutine* `sussingopen`
*subroutine* `sussingtreeindicesread`

*function* `root_finder_find`

*subroutine* `abundances_dump`
*subroutine* `chemicals_dump`
*subroutine* `history_dump`
*subroutine* `kepler_orbits_dump`

*subroutine* `merger_tree_data_set__subhalo_masses`
*subroutine* `agestatisticscreatelinked`
*subroutine* `blackholecreatelinked`

*subroutine* `diskcreatelinked`

*subroutine* `formationtimecreatelinked`
*subroutine* `hosthistorycreatelinked`
*subroutine* `interoutputcreatelinked`

*function* `root_finder_find`

*subroutine* `mergingstatisticscreatelinked`
*subroutine* `node_component__agestatistics_dump`
*subroutine* `node_component__agestatisticsnull_dump`
*subroutine* `node_component__agestatisticsstandard_dump`
*subroutine* `node_component__basicnonevolving_dump`
*subroutine* `node_component__basicstandard_dump`
*subroutine* `node_component__blackhole__dump`
*subroutine* `node_component__blackholesimple_dump`
*subroutine* `node_component__darkmatterprofile_dump`
*subroutine* `node_component__darkmatterprofilescalescale_dump`
*subroutine* `node_component__darkmatterprofilescalescalepreset_dump`

*subroutine* `node_component__basicnonevolving_dump`
*subroutine* `node_component__basicstandard_dump`
*subroutine* `node_component__blackholenull_dump`
*subroutine* `node_component__blackholestandard_dump`
*subroutine* `node_component__darkmatterprofilenull_dump`
*subroutine* `node_component__darkmatterprofilescalescalepreset_dump`
subroutine node_component_-darkmatterprofilescaleshape_dump
subroutine node_component_-diskexponential_dump
subroutine node_component_-diskverysimple_dump
subroutine node_component_-diskexponential_dump
subroutine node_component_-disknull_dump
subroutine node_component_-dynamicsstatistics_dump
subroutine node_component_-dynamicsstatisticsbars_dump
subroutine node_component_-formationtime_dump
subroutine node_component_-formationtimenull_dump
subroutine node_component_-hosthistorynull_dump
subroutine node_component_-hothalonull_dump
subroutine node_component_-hosthistorystandard_dump
subroutine node_component_-hotalooutflowtracking_dump
subroutine node_component_-hotalooutflowtracking_dump
subroutine node_component_-hotalooutflowtracking_dump
subroutine node_component_-hotaloeverysimple_dump
subroutine node_component_-indices_dump
subroutine node_component_-indicesnull_dump
subroutine node_component_-interoutputnull_dump
subroutine node_component_-interoutputstandard_dump
subroutine node_component_-massflowstatistics_dump
subroutine node_component_-massflowstatisticsnull_dump
subroutine node_component_-massflowstatisticsstandard_dump
subroutine node_component_-mergingstatisticsnull_dump
subroutine node_component_-mergingstatisticsstandard_dump
subroutine node_component_-mergingstatisticsrecent_dump
subroutine node_component_-nbodynull_dump
subroutine node_component_-nobodystandard_dump
subroutine node_component_-positionnull_dump
subroutine node_component_-positionpreset_dump
subroutine node_component_-satellitenull_dump
subroutine node_component_-satelliteorbiting_dump
subroutine node_component_-satellitepreset_dump
subroutine node_component_-satellitestandard_dump
19.1. Program units

subroutine node_component_spheroid_dump
subroutine node_component_spheroidnull_dump
subroutine node_component_spheroidstandard_dump
subroutine node_component_spin_dump
subroutine node_component_spinnull_dump
subroutine node_component_spinpreset3d_dump
subroutine node_component_spinnrandom_dump
subroutine node_dump_xml
subroutine satellitecreatelinked
subroutine spincreatelinked
subroutine tree_node_remove_from_mergee
subroutine node_component_disk_very_simple_post_evolve
subroutine stellar_luminosities_dump
subroutine tensor_r2_d3_sym_dump
function radiation_intergalactic_background_internal_update
subroutine satellite_merging_remnant_size_cole2000
function boylankolchin2008defaultconstructor
function jiang2008defaultconstructor
function nulldefaultconstructor
function villalobos2013defaultconstructor
function jiang2008defaultconstructor
function villalobos2013defaultconstructor
function infinitedefaultconstructor
function presetdefaultconstructor
function imf_energy_input_rate_noninstantaneous
function imf_recycling_rate_noninstantaneous
function stellar_population_luminosity
subroutine excursion_sets_first_crossing_rate_tabulate_farahi
function excursion_sets_first_crossing_probability_zhang_hui_high
subroutine node_component_spheroid_dump
subroutine node_component_spheroidnull_dump
subroutine node_component_spheroidstandard_dump
subroutine node_component_spinnull_dump
subroutine node_component_spinpreset3d_dump
subroutine node_component_spinnrandom_dump
subroutine positioncreatelinked
subroutine spheroidcreatelinked
subroutine treenode_remove_from_host
subroutine node_component_disk_expontential_post_evolve
subroutine node_component_spheroid_standard_post_evolve
subroutine tensor_r2_d3_sym_dump
subroutine tensor_r2_d3_sym_read_raw
subroutine satellite_merging_remnant_size_cole2000
function boylankolchin2008defaultconstructor
function infinitedefaultconstructor
function laceycole1993defaultconstructor
function presetdefaultconstructor
function wetzelwhite2010defaultconstructor
function laceycole1993defaultconstructor
function wetzelwhite2010defaultconstructor
function infinitedefaultconstructor
function nulldefaultconstructor
function presetdefaultconstructor
subroutine satellite_move_to_new_host
function imf_metal_yield_rate_noninstantaneous
subroutine mass_function_covariance_matrix
function excursion_sets_first_crossing_probability_farahi
function excursion_sets_first_crossing_probability_zhang_hui_high
function make_table
19. Source Code Documentation

subroutine transfer_function_file_read
subroutine input_parameters_file_open
subroutine alloc_array_character_1d_kind_int8
subroutine alloc_array_double_precision_1d_kind_int8
subroutine alloc_array_double_precision_1d
subroutine alloc_array_double_precision_2d
subroutine alloc_array_double_precision_2d_kind_int8
subroutine alloc_array_double_precision_3d
subroutine alloc_array_double_precision_3d_kind_int8
subroutine alloc_array_double_precision_4d
subroutine alloc_array_double_precision_4d_kind_int8
subroutine alloc_array_double_precision_5d
subroutine alloc_array_double_precision_5d_kind_int8
subroutine alloc_array_integer_1d_kind_int8
subroutine alloc_array_integer_2d_kind_int8
subroutine alloc_array_integer_kind_int8_1d
subroutine alloc_array_integer_kind_int8_2d
subroutine alloc_array_logical_1d_kind_int8
subroutine alloc_array_real_1d
subroutine dealloc_array_character_1d
subroutine dealloc_array_double_precision_1d
subroutine dealloc_array_double_precision_2d
subroutine dealloc_array_double_precision_4d
subroutine dealloc_array_double_precision_5d
subroutine dealloc_array_integer_1d
subroutine dealloc_array_integer_kind_int8_1d
subroutine dealloc_array_integer_kind_int8_2d
subroutine dealloc_array_logical_1d
subroutine dealloc_array_real_1d
subroutine memory_usage_record
subroutine unit_tests_finish

subroutine: create_indentation_format
Description: Create a format for indentation.
Code lines: 25
Contained by: module galacticus_display

subroutine: galacticus_display_counter
Description: Displays a percentage counter and bar to show progress.
Code lines: 11
Contained by: module galacticus_display
subroutine: galacticus_display_counter_clear
Description: Clears a percentage counter.
Code lines: 10
Contained by: module galacticus_display

subroutine: galacticus_display_counter_clear_lockless
Description: Clears a percentage counter.
Code lines: 18
Contained by: module galacticus_display

subroutine: galacticus_display_counter_lockless
Description: Displays a percentage counter and bar to show progress.
Code lines: 28
Contained by: module galacticus_display

interface: galacticus_display_indent
Code lines: 3
Contained by: module galacticus_display

subroutine: galacticus_display_indent_char
Description: Increase the indentation level and display a message.
Code lines: 34
Contained by: module galacticus_display

subroutine: galacticus_display_indent_varstr
Description: Increase the indentation level and display a message.
Code lines: 12
Contained by: module galacticus_display

interface: galacticus_display_message
Code lines: 3
Contained by: module galacticus_display

subroutine: galacticus_display_message_char
Description: Display a message (input as a character variable).
Code lines: 29
Contained by: module galacticus_display

subroutine: galacticus_display_message_varstr
Description: Display a message (input as a varying_string variable).
Code lines: 29
Contained by: module galacticus_display

subroutine: galacticus_display_unindent
Description: Decrease the indentation level and display a message.
Code lines: 34
Contained by: module galacticus_display
function: galacticus_verbosity_level
Description: Returns the verbositly level in GALACTICUS.
Code lines: 6
Contained by: module galacticus_display

subroutine: galacticus_verbosity_level_set
Description: Set the verbosity level.
Code lines: 7
Contained by: module galacticus_display

subroutine: initialize_display
Description: Initialize the module by determining the requested verbosity level.
Code lines: 24
Contained by: module galacticus_display

file: galacticus.error.F90
Description: Contains a module which implements error reporting for the GALACTICUS package.
Code lines: 203

module: galacticus_error
Description: Implements error reporting for the GALACTICUS package.
Code lines: 183
Contained by: file galacticus.error.F90
Modules used: fgs1 hdf5 semaphores
Used by: program conditional_mass_function program tests_excursion_sets program halo_mass_functions
program mass_function_covariance subroutine xermsg
function fgabnd function accretionhaloaccretedmassnull
function accretionhaloaccretedmasschemicalsnull
function accretionhaloaccretedmassmetalsnull
function accretionhaloaccretionratechemicalsnull
function accretionhaloaccretionratenull
function accretionhaloaccretionratemetalsnull
function accretionhaloaccretionratenull
function accretionhaloconstructornamed
function accretionhalofailedaccretedmassnull
function naozbarkana2007failedfraction
function simpleconstructor
function coldmodedefaultconstructor
subroutine accretion_disks_adaf_get_parameters
subroutine accretionhaloconstructor
function coldmodedefaultconstructor
subroutine accretion_disks_adaf_get_parameters
<table>
<thead>
<tr>
<th>Function/Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>accretiondiskspectraconstructor</td>
</tr>
<tr>
<td>accretiondiskspectraspectrumnull</td>
</tr>
<tr>
<td>subroutine fileloadfile</td>
</tr>
<tr>
<td>subroutine fileloadfile</td>
</tr>
<tr>
<td>function abundance_pattern_lookup</td>
</tr>
<tr>
<td>subroutine atomic_data_initialize</td>
</tr>
<tr>
<td>subroutine gaunt_factor_initialize</td>
</tr>
<tr>
<td>subroutine collisional_excitation_rate_initialize</td>
</tr>
<tr>
<td>subroutine atomic_rate_ionization_collisional_initial</td>
</tr>
<tr>
<td>subroutine dielectric_recombination_rate_initial</td>
</tr>
<tr>
<td>function atomic_rate_recombination_radiative</td>
</tr>
<tr>
<td>function atomic_rate_recombination_radiative_verner</td>
</tr>
<tr>
<td>function black_hole_binary_recoil_velocity</td>
</tr>
<tr>
<td>function black_hole_binary_separation_growth_rate_standard</td>
</tr>
<tr>
<td>function black_hole_frame_dragging_frequency_node</td>
</tr>
<tr>
<td>function black_hole_isco_radius_spin</td>
</tr>
<tr>
<td>function black_hole_metric_d_factor_node</td>
</tr>
<tr>
<td>subroutine chemical_hydrogen_rates_compute</td>
</tr>
<tr>
<td>subroutine chemical_state_cie_file_read</td>
</tr>
<tr>
<td>subroutine cooling_cold_mode_infall_rate_initialize</td>
</tr>
<tr>
<td>subroutine cooling_function_cie_file_read</td>
</tr>
<tr>
<td>subroutine cooling_radius_initialize</td>
</tr>
<tr>
<td>subroutine cooling_rate_initialize</td>
</tr>
<tr>
<td>subroutine cooling_rate modifier_cut_off</td>
</tr>
<tr>
<td>subroutine cooling_rate_simple_scaling_initialize</td>
</tr>
<tr>
<td>subroutine accretiondiskspectrainitialize</td>
</tr>
<tr>
<td>function fileconstructor</td>
</tr>
<tr>
<td>subroutine atomic_cross_section_ionization_photo_initial</td>
</tr>
<tr>
<td>function atom_lookup</td>
</tr>
<tr>
<td>subroutine atomic_ionization_potential_initial</td>
</tr>
<tr>
<td>subroutine gaunt_factor_sutherland1998</td>
</tr>
<tr>
<td>function collisional_excitation_cooling_rate</td>
</tr>
<tr>
<td>function dielectric_recombination_rate_arnaud1985</td>
</tr>
<tr>
<td>subroutine atomic_rate_recombination_radiative_initial</td>
</tr>
<tr>
<td>function black_hole_binary_initial_radius</td>
</tr>
<tr>
<td>function black_hole_binary_separation_growth_rate</td>
</tr>
<tr>
<td>subroutine black_hole_binary_merge</td>
</tr>
<tr>
<td>function black_hole_horizon_radius_node</td>
</tr>
<tr>
<td>function black_hole_metric_a_factor_node</td>
</tr>
<tr>
<td>function black_hole_static_radius_node</td>
</tr>
<tr>
<td>subroutine chemical_hydrogen_rates_initialize</td>
</tr>
<tr>
<td>subroutine chemical_state_initialize</td>
</tr>
<tr>
<td>subroutine cooling_cold_mode_infall_rate_dynamical_time_initial</td>
</tr>
<tr>
<td>subroutine cooling_function_initialize</td>
</tr>
<tr>
<td>subroutine cooling_radius_simple_initialize</td>
</tr>
<tr>
<td>subroutine cooling_rate_white_frenk_initial</td>
</tr>
<tr>
<td>subroutine cooling_rate_simple_initialize</td>
</tr>
<tr>
<td>subroutine cooling_time_initialize</td>
</tr>
</tbody>
</table>
19. Source Code Documentation

subroutine freefall_radius_initialize
subroutine freefall_time_available_halo_formation_initialize
subroutine cooling_specific_angular_momentum_initialize
subroutine cooling_time_available_halo_formation_initialize
subroutine cooling_time_available_halo_formation_initialize
function cosmologyfunctionsconstructornamed
function cosmologyfunctionsconstructormean
function cosmologyfunctionsdensityscalingearlytime
function cosmologyfunctionsdistanceangular
function cosmologyfunctionsdistancecomovingconvert
function cosmologyfunctionsdistancecomovingnull
function cosmologyfunctionsdistanceluminositynull
function cosmologyfunctionsdominationepochmatternull
function cosmologyfunctionsexpansionfactorisvalidnull
function cosmologyfunctionsexpansionfactornull
function cosmologyfunctionsexpansionratenull
function cosmologyfunctionsomegadarkenergynull
function cosmologyfunctionsomegamatterepochalnull
function cosmologyfunctionstemperaturecmbepochal
function matterdarkenergycosmictime
function matterdarkenergycosmictimeatdistancecomovingnull
function matterdarkenergyequationofstatedarkenergy
function matterdarkenergyexponentdarkenergy
function matterdarkenergyhubbleparameterepochal
function matterdarkenergymegadarkenergypochalnull
function matterdarkenergytimeatdistancecomovingnull
function matterdarkenergytimeatdistancecomoving
19. Source Code Documentation

function omegadarkenergysimple
function omegamattersimple
subroutine dark_matter_mass_accretion_time_zhao2009
subroutine dark_matter_halo_mass_accretion_time_zhao2009
function darkmatterhaloscaleconstructornamed
function darkmatterhaloscaleinitialize
function darkmatterhaloscaledynamictimescalenull
subroutine darkmatterhaloscaleinitialize
function darkmatterhaloscalemeandensitygrowthrate
function darkmatterhaloscalemeandensitynull
function darkmatterhaloscalevirialradiusgrowthrate
function darkmatterhaloscalevirialradiusnull
function darkmatterhaloscalevirialtemperaturenull
function darkmatterhaloscalevirialvelocitygrowthrate
function darkmatterhaloscalevirialvelocitynull
subroutine dark_matter_halo_spins_initialize
function dark_matter_halo_spins_initialize
function halo_spin_distribution_sample
function einastoconstructor
function darkmatterprofilecircularvelocitymaximumnull
function darkmatterprofilecircularvelocitynull
function darkmatterprofileconstructornamed
function darkmatterprofileenclosedmassnull
function darkmatterprofileenergygrowthratenull
function darkmatterprofilefreefallradiusincreaseratenull
function darkmatterprofilefreefallradiusnull
function darkmatterprofilekspacenull
function darkmatterprofilepotentialnull
function darkmatterprofileradiusfromspecificangularmomentumnull
function darkmatterprofilerotationnormalizationnull
function einastoconstructor
function isothermalpotential
function nfwconstructor
function diemerkravtsov2014constructor
function duttonmaccio2014constructor
function darkmatterprofileconcentrationconcentrationnull
function darkmatterprofileconcentrationdarkmatterprofiledefinitionnull
function darkmatterprofileconcentrationdensitycontrastdefinitionnull
subroutine darkmatterprofileconcentrationinitialize
function diemerkravtsov2014constructor
function duttonmaccio2014constructor
subroutine dark_matter_shapes_initialize
function dark_matter_profile_potential_task
subroutine galactic_dynamics_bar_instability_initialize
subroutine events_node_merger_do_slh
subroutine galactic_structure_density
subroutine galactic_structure_enclosed_mass_defaults
19.1. Program units

function galactic_structure_radius_enclosing_mass
function galactic_structure_mass_type_decode
subroutine galactic_structure_radii_solve
subroutine galactic_structure_radii_solveadiabatic
subroutine galactic_structure_radius_initial_initialize
function galactic_structure_surface_density
subroutine galacticus_time_per_tree_file_initialize
subroutine galacticus_output_analysis_mass_dpndnt_sz_dstrbtins
subroutine load_primus_mass_function
function galacticus_build_string
function filter_lightcone_get_coordinates
subroutine galacticus_merger_tree_lightcone_geometry_initialize
subroutine galacticus_output_tree_descendants_initialize
subroutine galacticus_output_tree_final_descendants_initialize
subroutine galacticus_output_tree_satellite_status_initialize
subroutine galacticus_output_star_formation_histories_initialize
subroutine star_formation_history_metallicity_split_make_history
subroutine galacticus_output_tree_velocity_dispersion_initialize
subroutine galacticus_version_output
subroutine geometry_survey_window_functions_caputi_2011_ukidss_uds
subroutine geometry_survey_window_functions_li_white_2009_sdss
subroutine conditional_mass_functions_initialize
function betaprofiledefaultconstructor

function hothalomassdistributiondensitylogslopenull
function hothalomassdistributionenclosedmassnull
function hothalomassdistributionradialmomentnull
<table>
<thead>
<tr>
<th>Function/Function Name</th>
<th>Subroutine/Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ricotti2000defaultconstructor</td>
<td>ricotti2000defaultconstructor</td>
</tr>
<tr>
<td>betaprofiledefaultconstructor</td>
<td>growingconstructor</td>
</tr>
<tr>
<td>hothalomassdistributioncoreradiusconstructormed</td>
<td>growingconstructor</td>
</tr>
<tr>
<td>hothalomassdistributioncoreradiusinitialize</td>
<td>hot_halo_ram_pressure_force</td>
</tr>
<tr>
<td>hot_halo_ram_pressure_radiusradiusnull</td>
<td>hot_halo_ram_pressure_stripping_radius</td>
</tr>
<tr>
<td>hot_halo_ram_pressure_stripping_timescale</td>
<td>hot_halo_temperature_stripping_radiusinitialize</td>
</tr>
<tr>
<td>filter_response_load</td>
<td>filter_vega_offset</td>
</tr>
<tr>
<td>filereaddata</td>
<td>intergalacticmediumstateelectronfractional</td>
</tr>
<tr>
<td>intergalacticmediumstateelectronscatteringopticaldepth</td>
<td>intergalacticmediumstateelectronscattering</td>
</tr>
<tr>
<td>intergalacticmediumstateelectronscatteringinitialize</td>
<td>intergalacticmediumstateelectronscattering</td>
</tr>
<tr>
<td>intergalacticmediumstateneutralheliumfractionnull</td>
<td>intergalacticmediumstateneutralheliumfractionnull</td>
</tr>
<tr>
<td>intergalacticmediumstatesinglyionizedheliumneutralhelium</td>
<td>intergalacticmediumstatetemperaturenull</td>
</tr>
<tr>
<td>intergalacticmediumstateelectron_fractional</td>
<td>intergalacticmediumstateelectron_fractional</td>
</tr>
<tr>
<td>internal_update</td>
<td>inverse_gamma_function_incomplete_complementary</td>
</tr>
<tr>
<td>generalized_press_schechter Branch_mass</td>
<td>tree_branching_initialize</td>
</tr>
<tr>
<td>tree_branching_modifiers_initialize</td>
<td>generalized_press_schechter_branch_mass</td>
</tr>
<tr>
<td>tree_branching_modifiers_initialize</td>
<td>tree_branching_modifiers_initialize</td>
</tr>
<tr>
<td>merger_tree_create</td>
<td>merger_tree_create</td>
</tr>
<tr>
<td>merger_tree_build_initialize</td>
<td>merger_tree_build_initialize</td>
</tr>
<tr>
<td>merger_tree_construct_fully_specified</td>
<td>merger_tree_construct_fully_specified</td>
</tr>
<tr>
<td>node_lookup</td>
<td>node_lookup</td>
</tr>
<tr>
<td>assign_mergers</td>
<td>assign_mergers</td>
</tr>
<tr>
<td>assign_scale_radii</td>
<td>assign_scale_radii</td>
</tr>
<tr>
<td>build_isolated_parent_pointers</td>
<td>build_isolated_parent_pointers</td>
</tr>
<tr>
<td>build_subhalo_mass_histories</td>
<td>build_subhalo_mass_histories</td>
</tr>
<tr>
<td>merger_tree_read_do</td>
<td>merger_tree_read_do</td>
</tr>
<tr>
<td>scan_for_mergers</td>
<td>scan_for_mergers</td>
</tr>
</tbody>
</table>
19.1. Program units

function galacticusangularmomentaincludesubhalos

subroutine galacticusimport

subroutine galacticusopen

function galacticussubhalotracecount

function galacticustreeweight

function importerunitconvert1d

function importerunitconvert2d

function importerunitconvertscalar

function mergertreeimporterangularmomenta3davailablenull

function mergertreeimporterangularmomentaavailablenull

function mergertreeimporterangularmomentaincludesubhalosnull

function mergertreeimporterclosenull

function mergertreeimporterconstructornamed

subroutine mergertreeimporterimportnull

function mergertreeimportermassesincludesubhalosnull

subroutine mergertreeimporternodecountnull

function mergertreeimporterparticlecountavailablenull

function mergertreeimporterpositionsareperiodicnull

function mergertreeimporterpositionsavailablenull

function mergertreeimporterscalerradiiavailablenull

function mergertreeimporterspin3davailablenull

function mergertreeimporterspinavailablenull

subroutine mergertreeimportersubhalotracecountnull

function mergertreeimportertreecountnull

function mergertreeimportertreecountnull

function mergertreeimportertreeshavesubhalosnull

function mergertreeimportertreeweightnull

function mergertreeimportertreevelocitiesincludehubbleflowavailablenull

function mergertreeimportertreevelocitiesincludehubbleflew

function sussingangularmomentaincludesubhalos

subroutine sussingopen

subroutine sussingsubhalotrace

function sussingdefaultconstructor

function sussingmassesincludesubhalos

subroutine sussingreadhaloascii

function sussingtreeindicesread

function sussingtreeweight

function sussingangularmomentaincludesubhalos

function sussingdefaultconstructor

function sussingmassesincludesubhalos

subroutine sussingreadhaloascii

subroutine sussingtreeindicesread

function sussingtreeweight

function galacticusangularmomentaincludesubhalos
function galacticussubbelength
function galacticusmassesincludesubhalos
subroutine galacticussubhalotrace
subroutine galacticustreeindicesread
subroutine merger_tree_state_restore
subroutine merger_tree_evolve_to
subroutine tree_node_evolve_initialize
subroutine merger_tree_mass_accretion_history_output
subroutine ode_solve
function integrate
function interpolate_derivative
subroutine meshes_apply_point
function root_finder_find
function search_array_for_closest
subroutine abundances_builder
function abundances_names
subroutine chemicals_builder
function chemical_database_get_index
subroutine coordinates_null_from
function history_add
subroutine history_combine
subroutine history_extend
subroutine history_long_integer_builder
subroutine history_long_integer_trim
function coordinates_null_to
function integrate
function interpolate
function interpolate_derivative
function make_range
function root_finder_range_expand
function abundances_atomic_index
function abundances_get_metallicity
function abundances_set_metallicity
function chemicals_names
function chemical_structure_initialize
function coordinates_null_to
function history_builder
subroutine history_create
subroutine history_increment
subroutine history_long_integer_create
function history_subtract
subroutine kepler_orbits Assert_is_defined
function kepler_orbits_host_mass
function kepler_orbits_radius
function kepler_orbits_velocity_radial
function kepler_orbits_velocity_tangential
function mass_distribution_beta_profile_initiaize
function mass_distribution_density_gradient_radial_null
function mass_distribution_density_radial_moment_null
function mass_distribution_hernquist_density_radial_moment
function mass_distribution_create
function mass_distribution_density_null
function mass_distribution_half_mass_radius_spherical
subroutine mass_distribution_
19.1. Program units

function mass_distribution_mass_enc_by_sphere_null
function mass_distribution_potential_null
subroutine mass_distribution_sersic_initialize
subroutine merger_tree_data_structure_add_metadata
subroutine merger_tree_data_structure_export
subroutine merger_tree_data_structure_read_particles_ascii
subroutine merger_tree_data_structure_set_property_integer8
subroutine merger_tree_data_validate_trees
subroutine agestatisticsdiskintegratedsfrrate
function agestatisticsget
subroutine agestatisticsdisktimeweightedintegratedsfrrate
subroutine agestatistsspheroidintegratedsfrrate
function basicget
function blackholeget
subroutine diskabundancesgasrate
subroutine diskangularmomentumrate
subroutine diskexponentialmassgasrategeneric
function diskget
function diskmassgasrate
function dynamicsstatisticsget
function hosthistoryget
subroutine hothaloabundancescoldrate
subroutine hothaloangularmomentumcoldrate
subroutine hothalochemicalsrate
subroutine hothalocoldmodeoutflowreturn
function hothaloget
subroutine hothalomassrate
subroutine hothalocoldmodeouterradiusgrowthrate
function hothalostandardouterradiusgrowthrate
subroutine hothalounaccretedmassrate
function indicesget
function interoutputget
function massflowstatisticsget
function nbodyget

subroutine node_component_-_agestatistics_builder
subroutine node_component_-_agestatistics_initializer
subroutine node_component_-_agestatistics_remove
subroutine node_component_-_agestatisticsstandard_builder
subroutine node_component_basic_-_initializer
subroutine node_component_basic_remove
subroutine node_component_basicnull_builder
subroutine node_component_basicstandard_builder
subroutine node_component_basicstandardtracking_builder
subroutine node_component_blackhole_-_builder
subroutine node_component_blackhole_-_initializer
subroutine node_component_blackhole_remove
subroutine node_component_blackhole_blackholenull_builder
subroutine node_component_blackhole_blackholesimple_builder
subroutine node_component_blackhole_blackholestandard_builder
subroutine node_component_blackhole_blackholestandardtracking_builder
subroutine node_component_blackhole_blackholestandardtracking_move
subroutine node_component_blackhole_blackholeblackhole_standard_builder
subroutine node_component_blackhole_blackholeblackhole_standard_move
subroutine node_component_blackhole_blackholeblackholevery_simple_builder
subroutine node_component_blackhole_blackholeblackholevery_simple_remove
subroutine node_component_blackhole_blackholeblackholevery_simple_blackholenull_builder
subroutine node_component_blackhole_blackholeblackholevery_simple_diskexponential_builder
subroutine node_component_blackhole_blackholeblackholevery_simple_move

subroutine node_component_disk_-_exponential_attach_pipes
subroutine node_component_disk_-_initializer
subroutine node_component_disk_remove
subroutine node_component_disk_very_very_simple_attach_pipe
subroutine node_component_disk_very_very_simple_enclosed_mass

subroutine interoutputspheroidstarformationraterate
function mergingstatisticsget
subroutine node_component_-_node_component_blackhole_blackholeblackhole_standard_builder
subroutine node_component_basic_move
subroutine node_component_basicnull_builder
subroutine node_component_basicstandard_builder
subroutine node_component_basicstandardtracking_builder
subroutine node_component_blackhole_blackholestandard_builder
subroutine node_component_blackhole_blackholestandardtracking_builder
subroutine node_component_blackhole_blackholeblackhole_standard_builder
subroutine node_component_blackhole_blackholeblackholestandardtracking_builder
subroutine node_component_blackhole_blackholeblackholevery_simple_builder
subroutine node_component_blackhole_blackholeblackholevery_simple_remove
subroutine node_component_blackhole_blackholeblackholevery_simple_blackholenull_builder
subroutine node_component_blackhole_blackholeblackholevery_simple_diskexponential_builder
subroutine node_component_blackhole_blackholeblackholevery_simple_move

subroutine node_component_disk_diskexponential_builder
subroutine node_component_disk_diskverysimple_builder
subroutine node_component_disk_dynamicsstatistics_builder
subroutine node_component_disk_dynamicsstatistics_initializor
19.1. Program units

subroutine node_component_-dynamicsstatistics_move
subroutine node_component_-dynamicsstatistics_remove
subroutine node_component_-dynamicsstatisticsbars_builder
subroutine node_component_-formationtime_builder
subroutine node_component_-formationtime_move
subroutine node_component_-formationtime_remove
subroutine node_component_-formationtimecole2000_builder
subroutine node_component_hosthistory_-builder
subroutine node_component_hothalo_-standard_mass_removal_rate_builder
subroutine node_component_indices_-builder
subroutine node_component_interoutput_-builder
subroutine node_component_massflowstatistics_builder
subroutine node_component_massflowstatistics_move
subroutine node_component_massflowstatisticsnull_builder
subroutine node_component_massflowstatisticsstandard_builder

subroutine node_component_-dynamicsstatisticsnull_builder
subroutine node_component_-formationtime_initializor
subroutine node_component_-formationtimenull_builder
subroutine node_component_hothistory_-initializor
subroutine node_component_hothistorynull_builder
subroutine node_component_hothalo_-initialize
subroutine node_component_hothalo_-move
subroutine node_component_hothalo_-remove
subroutine node_component_hothalonull_-builder
subroutine node_component_hothalo_outflowtracking_builder
subroutine node_component_hothalo_standard_builder
subroutine node_component_hothalo_very_simple_builder
subroutine node_component_indices_-initialize
subroutine node_component_indices_-move
subroutine node_component_indicesnull_-builder
subroutine node_component_interoutputnull_-builder
subroutine node_component_interoutputnull_-move
subroutine node_component_interoutputnull_-remove
subroutine node_component_interoutputstandard_builder
subroutine node_component_interoutput_very_simple_builder
subroutine node_component_interoutput_initiator
subroutine node_component_interoutputnull_builder
subroutine node_component_interoutputstandard_builder
subroutine node_component_interoutputnull_builder
subroutine node_component_massflowstatistics_initiator
subroutine node_component_massflowstatisticsnull_builder
subroutine node_component_massflowstatisticsstandard_builder
<table>
<thead>
<tr>
<th>subroutine</th>
<th>subroutine</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_component_-</td>
<td>node_component_-</td>
</tr>
<tr>
<td>mergingstatistics_builder</td>
<td>mergingstatistics_initializor</td>
</tr>
<tr>
<td>node_component_-</td>
<td>node_component_-</td>
</tr>
<tr>
<td>mergingstatistics_move</td>
<td>mergingstatistics_remove</td>
</tr>
<tr>
<td>node_component_-</td>
<td>node_component_-</td>
</tr>
<tr>
<td>mergingstatisticsnull_builder</td>
<td>mergingstatisticsrecent_builder</td>
</tr>
<tr>
<td>node_component_-</td>
<td>node_component_nbody_builder</td>
</tr>
<tr>
<td>mergingstatisticsstandard_builder</td>
<td>node_component_nbody_initializor</td>
</tr>
<tr>
<td>node_component_nbody_move</td>
<td>node_component_nbody_remove</td>
</tr>
<tr>
<td>node_component_nbody_null_builder</td>
<td>node_component_nbody_null_builder</td>
</tr>
<tr>
<td>node_component_nbody_initializor</td>
<td>node_component_nbody_initializor</td>
</tr>
<tr>
<td>node_component_nbody_position_-</td>
<td>node_component_nbody_position_move</td>
</tr>
<tr>
<td>node_component_nbody_position_remove</td>
<td>node_component_nbody_position_remove</td>
</tr>
<tr>
<td>node_component_position_builder</td>
<td>node_component_position_builder</td>
</tr>
<tr>
<td>node_component_position_move</td>
<td>node_component_position_move</td>
</tr>
<tr>
<td>node_component_position_null_builder</td>
<td>node_component_position_null_builder</td>
</tr>
<tr>
<td>node_component_position_preset_builder</td>
<td>node_component_position_preset_builder</td>
</tr>
<tr>
<td>node_component_satellite_builder</td>
<td>node_component_satellite_builder</td>
</tr>
<tr>
<td>node_component_satellite_move</td>
<td>node_component_satellite_move</td>
</tr>
<tr>
<td>node_component_satellite_null_builder</td>
<td>node_component_satellite_null_builder</td>
</tr>
<tr>
<td>node_component_satellite_orbiting_builder</td>
<td>node_component_satellite_orbiting_builder</td>
</tr>
<tr>
<td>node_component_satellite_orbiting</td>
<td>node_component_satellite_orbiting</td>
</tr>
<tr>
<td>node_component_satellite_orbiting_builder</td>
<td>node_component_satellite_orbiting_builder</td>
</tr>
<tr>
<td>node_component_satellite_preset_builder</td>
<td>node_component_satellite_preset_builder</td>
</tr>
<tr>
<td>node_component_satellite_preset</td>
<td>node_component_satellite_preset</td>
</tr>
<tr>
<td>node_component_satellite_random_builder</td>
<td>node_component_satellite_random_builder</td>
</tr>
<tr>
<td>node_component_satellite_random</td>
<td>node_component_satellite_random</td>
</tr>
<tr>
<td>node_component_satellite_random_builder</td>
<td>node_component_satellite_random_builder</td>
</tr>
<tr>
<td>node_component_satellite_random_preset_builder</td>
<td>node_component_satellite_random_preset_builder</td>
</tr>
<tr>
<td>node_component_satellite_random_preset</td>
<td>node_component_satellite_random_preset</td>
</tr>
<tr>
<td>node_component_satellite_random_preset Builder</td>
<td>node_component_satellite_random_preset Builder</td>
</tr>
<tr>
<td>node_component_spheroid_builder</td>
<td>node_component_spheroid_builder</td>
</tr>
<tr>
<td>node_component_spheroid_move</td>
<td>node_component_spheroid_move</td>
</tr>
<tr>
<td>node_component_spheroid_null_builder</td>
<td>node_component_spheroid_null_builder</td>
</tr>
<tr>
<td>node_component_spheroid_preset_builder</td>
<td>node_component_spheroid_preset_builder</td>
</tr>
<tr>
<td>node_component_spheroid_preset</td>
<td>node_component_spheroid_preset</td>
</tr>
<tr>
<td>node_component_spheroid_preset_preset_builder</td>
<td>node_component_spheroid_preset_preset_builder</td>
</tr>
<tr>
<td>node_component_spheroid_preset_preset</td>
<td>node_component_spheroid_preset_preset</td>
</tr>
<tr>
<td>function positionget</td>
<td>function spheroidabundancesgasrate</td>
</tr>
<tr>
<td>subroutine spheroidabundancesgasrate</td>
<td>subroutine spheroidabundancesstarrate</td>
</tr>
</tbody>
</table>
subroutine spheroidangularmomentumrate
subroutine spheroidmassgasrate
function spheroidget
subroutine spheroidluminositiesstellarrate
subroutine spheroidmassstellarrate
subroutine spheroidstandardabundancesgasrategeneric
subroutine spheroidstandardabundancesstellarrategeneric
subroutine spheroidstandardluminositiesstellarrategeneric
subroutine spheroidstandardmassstellarrategeneric
subroutine spheroidstandardstellarpropertieshistoryrategeneric
subroutine spheroidstarformationhistoryrate
subroutine spheroidstellarpropertieshistoryrate
function spinget
subroutine treenodeinitialize

subroutine node_component_basic_non_evolving_promote
subroutine node_component_basic_standard_tracking_promote
subroutine node_component_dark_matter_profile_scale_preset_promote
subroutine node_component_disk_exponential_post_evolve
subroutine node_component_disk_exponential_satellite_merging
subroutine node_component_hot_halo_standard_heat_source
subroutine node_component_hot_halo_standard_mass_sink
subroutine node_component_inter_output_standard_satellite_merger
subroutine node_component_merging_statistics_recent_node_merger
subroutine node_component_spheroid_standard_initialise
subroutine node_component_spheroid_standard_satellite_merging
subroutine node_component_spheroid_standard_preset_promote
subroutine node_component_spheroid_standard_random_promote
subroutine node_component_spin_preset_promote
subroutine node_component_spin_random_promote

1105
<table>
<thead>
<tr>
<th>Function/Program Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source Code Documentation</strong></td>
<td></td>
</tr>
<tr>
<td><code>function stellar_luminosities_index</code></td>
<td></td>
</tr>
<tr>
<td><code>function stellar_luminosities_is_output</code></td>
<td></td>
</tr>
<tr>
<td><code>function stellar_luminosities_name</code></td>
<td></td>
</tr>
<tr>
<td><code>subroutine table_1d_reverse</code></td>
<td></td>
</tr>
<tr>
<td><code>subroutine table_generic_1d_populate_single</code></td>
<td></td>
</tr>
<tr>
<td><code>subroutine table_linear_1d_populate_single</code></td>
<td></td>
</tr>
<tr>
<td><code>subroutine table_linear_cspline_1d_populate_single</code></td>
<td></td>
</tr>
<tr>
<td><code>subroutine tensor_r2_d3_sym_builder</code></td>
<td></td>
</tr>
<tr>
<td><code>subroutine tensor_r2_d3_sym_from_matrix</code></td>
<td></td>
</tr>
<tr>
<td><code>program optimal_sampling_smf</code></td>
<td></td>
</tr>
<tr>
<td><code>subroutine radiation_igb_file_initialize</code></td>
<td></td>
</tr>
<tr>
<td><code>function radiation_intergalactic_background_internal_update</code></td>
<td></td>
</tr>
<tr>
<td><code>function ram_pressure_stripping_mass_loss_rate_disk</code></td>
<td></td>
</tr>
<tr>
<td><code>subroutine satellite_dynamical_friction_acceleration_initialize</code></td>
<td></td>
</tr>
<tr>
<td><code>subroutine satellite_merging_mass_movement</code></td>
<td></td>
</tr>
<tr>
<td><code>subroutine satellite_merging_remnant_size_cole2000</code></td>
<td></td>
</tr>
<tr>
<td><code>subroutine satellite_merging_remnant_size</code></td>
<td></td>
</tr>
<tr>
<td><code>subroutine satellite_merging_remnant_progenitor_properties_cole2000</code></td>
<td></td>
</tr>
<tr>
<td><code>subroutine satellite_merging_remnant_progenitor_properties_standard</code></td>
<td></td>
</tr>
<tr>
<td><code>function boylankolchin2008timeuntilmerging</code></td>
<td></td>
</tr>
<tr>
<td><code>function jiang2008defaultconstructor</code></td>
<td></td>
</tr>
<tr>
<td><code>function jiang2008timeuntilmergingnull</code></td>
<td></td>
</tr>
<tr>
<td><code>function jiang2008timeuntilmerging</code></td>
<td></td>
</tr>
<tr>
<td><code>function virial_orbital_parameters_benson2005</code></td>
<td></td>
</tr>
<tr>
<td><code>function equivalent_circular_orbit_solver</code></td>
<td></td>
</tr>
</tbody>
</table>
19.1. Program units

subroutine satellite_orbit_extremum_phase_space_coordinates
subroutine satellite_tidal_heating_rate_initialize
function imf_descriptor

function imf_metal_yield_rate_noninstantaneous
function imf_recycling_rate_noninstantaneous
subroutine star_formation_imf_initialize_piecewisepowerlaw
function imf_index_lookup

subroutine star_formation_feedback_spheroids_initialize
subroutine star_formation_expulsive_feedback_spheroids_initialize
subroutine star_formation_rate_surface_density_disks_initialize
subroutine star_formation_rate_surface_density_disks_br_initialize
subroutine star_formation_timescale_disks_dynamical_time_initialize
subroutine star_formation_timescale_disks_mass_function_coevariance_matrix
subroutine star_formation_feedback_initialize

subroutine supernovae_population_iii_initialize
subroutine supernovae_type_ia_initialize
subroutine stellar_tracks_initialize

subroutine stellar_winds_initialize
subroutine stellar_population_properties_rates_initialize
function stellar_population_spectra_file_interpolate
subroutine spectrapostprocessorapplynull
subroutine spectrapostprocessorinitialize
function critical_overdensity_for_collapse
subroutine critical_overdensity_initialize

function satellite_tidal_field
subroutine satellite_tidal_stripping_initialize
function imf_energy_input_rate_noninstantaneous
function imf_name

subroutine star_formation_imf_initialize_piecewisepowerlaw
function imf_select_disk_spheroid

subroutine star_formation_feedback_disks_initialize
subroutine star_formation_expulsive_feedback_disks_initialize
subroutine star_formation_rate_surface_density_disks_initialize
subroutine star_formation_timescale_disks_dynamical_time_initialize
subroutine star_formation_timescale_spheroids_initialize
subroutine stellar_astrophysics_initialize

subroutine stellar_astrophysics_file_initialize
subroutine supernovae_population_iii_hegerwoosley_initialize
subroutine supernovae_type_ia_nagashima_initialize
subroutine stellar_tracks_initialize_file
function stellar_population_luminosity
subroutine stellar_population_spectrum_initialize
subroutine stellar_population_spectra_file_read
function spectrapostprocessorconstructormnamed
subroutine spectrapostprocessorconstructortonamed
function critical_overdensity_for_collapse_time_gradient
subroutine critical_overdensity_mass_scaling_initialize
19. Source Code Documentation

subroutine critical_overdensity_kitayama_suto1996_initialize
subroutine excursion_sets_barrier_initialize
function excursion_sets_first_crossing_rate_zhang_hui
function excursion_sets_first_crossing_rate_zhang_hui_high
subroutine dark_matter_halo_bias_initialize
subroutine halo_mass_function_tinker2008_initialize
function linear_growth_factor_logarithmic_derivative
subroutine initialize_cosmological_mass_variance
function power_spectrum_nonlinear_cosmicemu
function power_spectrum_nonlinear_peacockdodds1996
subroutine power_spectrum_window_functions_initialize
subroutine make_table
function transfer_function_half_mode_mass_null
subroutine transfer_function_initialize
function bryannorman1998defaultconstructor
function fixeddefaultconstructor
function sphericalcollapse_matterlambda_densitycontrast
function sphericalcollapse_matterlambda_densitycontrast_rateofchange
function virial_densitycontrast_densitycontrast_null
function virial_densitycontrast_densitycontrast_rateofchange
subroutine system_command_do
19.1. Program units

program test_nodes

function tidal_stripping_mass_loss_rate_disk

subroutine io_hdf5_assert_dataset_type

function tidal_stripping_mass_loss_rate_spheroid

subroutine io_hdf5_close

subroutine io_hdf5_create_reference_scalar_to_2d

function io_hdf5_dataset_rank

function io_hdf5_create_reference_scalar_to_4d

function io_hdf5_has_attribute

function io_hdf5_has_group

function io_hdf5_is_reference

function io_hdf5_open_dataset

function io_hdf5_open_group

subroutine io_hdf5_read_attribute_character_1d_array_static

subroutine io_hdf5_read_attribute_double_1d_array_allocatable

subroutine io_hdf5_read_attribute_double_scalar

subroutine io_hdf5_read_attribute_integer8_1d_array_allocatable

subroutine io_hdf5_read_attribute_integer_scalar

subroutine io_hdf5_read_attribute_varstring_1d_array_allocatable

subroutine io_hdf5_read_attribute_character_1d_array_allocatable

subroutine io_hdf5_read_attribute_double_1d_array_static

subroutine io_hdf5_read_attribute_double_2d_array_allocatable

subroutine io_hdf5_read_attribute_double_3d_array_allocatable

subroutine io_hdf5_read_dataset_character_1d_array_allocatable

subroutine io_hdf5_read_dataset_double_1d_array_allocatable

subroutine io_hdf5_read_dataset_double_2d_array_allocatable

subroutine io_hdf5_read_dataset_double_3d_array_allocatable

subroutine io_hdf5_read_dataset_double_4d_array_allocatable

subroutine io_hdf5_read_dataset_double_5d_array_allocatable

function io_hdf5_dataset_size

function io_hdf5_has_dataset

subroutine io_hdf5_initialize

function io_hdf5_open_attribute

function io_hdf5_open_dataset

subroutine io_hdf5_open_file

function io_hdf5_open_group

subroutine io_hdf5_read_attribute_character_1d_array_allocatable

subroutine io_hdf5_read_attribute_character_scalar

subroutine io_hdf5_read_attribute_double_1d_array_allocatable

subroutine io_hdf5_read_attribute_double_scalar

subroutine io_hdf5_read_attribute_integer8_1d_array_allocatable

subroutine io_hdf5_read_attribute_integer8_scalar

subroutine io_hdf5_read_attribute_integer_1d_array_allocatable

subroutine io_hdf5_read_attribute_integer_scalar

subroutine io_hdf5_read_attribute_varstring_1d_array_allocatable

subroutine io_hdf5_read_attribute_varstring_scalar

subroutine io_hdf5_read_attribute_character_1d_array_static

subroutine io_hdf5_read_attribute_character_scalar

subroutine io_hdf5_read_attribute_double_1d_array_static

subroutine io_hdf5_read_attribute_double_scalar

subroutine io_hdf5_read_attribute_integer8_1d_array_static

subroutine io_hdf5_read_attribute_integer8_scalar

subroutine io_hdf5_read_attribute_integer_1d_array_static

subroutine io_hdf5_read_attribute_varstring_1d_array_static

subroutine io_hdf5_read_attribute_varstring_scalar

subroutine io_hdf5_read_dataset_character_1d_array_static

subroutine io_hdf5_read_dataset_double_1d_array_static

subroutine io_hdf5_read_dataset_double_2d_array_static

subroutine io_hdf5_read_dataset_double_3d_array_static

subroutine io_hdf5_read_dataset_double_4d_array_static

subroutine io_hdf5_read_dataset_double_5d_array_static

1109
subroutine io_hdf5_read_dataset_integer8_1d_array_allocatable
subroutine io_hdf5_read_dataset_integer8_1d_array_static
subroutine io_hdf5_read_dataset_integer_1d_array_allocatable
subroutine io_hdf5_read_dataset_integer_1d_array_static
subroutine io_hdf5_read_dataset_varstring_1d_array_allocatable
subroutine io_hdf5_read_dataset_varstring_1d_array_static
subroutine io_hdf5_set_defaults
subroutine io_hdf5_uninitialize
subroutine io_hdf5_write_attribute_character_1d
subroutine io_hdf5_write_attribute_character_scalar
subroutine io_hdf5_write_attribute_double_1d
subroutine io_hdf5_write_attribute_double_scalar
subroutine io_hdf5_write_attribute_integer8_1d
subroutine io_hdf5_write_attribute_integer8_scalar
subroutine io_hdf5_write_attribute_integer_1d
subroutine io_hdf5_write_attribute_integer_scalar
subroutine io_hdf5_write_dataset_character_1d
subroutine io_hdf5_write_dataset_double_1d
subroutine io_hdf5_write_dataset_double_2d
subroutine io_hdf5_write_dataset_double_3d
subroutine io_hdf5_write_dataset_double_4d
subroutine io_hdf5_write_dataset_double_5d
subroutine io_hdf5_write_dataset_integer8_1d
subroutine io_hdf5_write_dataset_integer_1d
subroutine io_hdf_assert_is_initialized
subroutine xml_extrapolation_element_decode
function xml_get_first_element_by_tag_name
subroutine array_which
function count_lines_in_file_char
function value_integer_scalar_vs
function hash_perfect_index
function hash_perfect_is_present
function hash_perfect_size
module input_parameters
module memory_management
subroutine assert_logical_scalar

function: galacticus_component_list
Description: Construct a message describing which implementations of a component class provide required functionality.
Code lines: 15
Contained by: module galacticus_error
Modules used: iso_varying_string string_handling

subroutine: galacticus_error_handler_register
Description: Register signal handlers.
Code lines: 13
Contained by: module galacticus_error
Modules used: fgsl

interface: galacticus_error_report
19.1. Program units

Code lines: 3
Contained by: module galacticus_error

subroutine: galacticus_error_report_char
Description: Display an error message (optionally reporting the unit name in which the error originated) and stop.
Code lines: 20
Contained by: module galacticus_error

subroutine: galacticus_error_report_varstr
Description: Display an error message.
Code lines: 10
Contained by: module galacticus_error

Modules used: iso_varying_string

subroutine: galacticus_gsl_error_handler
Description: Handle errors from the GSL library, by flushing all data and then aborting.
Code lines: 25
Contained by: module galacticus_error
Modules used: fgsl iso_c_binding

subroutine: galacticus_signal_handler_sigfpe
Description: Handle SIGFPE signals, by flushing all data and then aborting.
Code lines: 18
Contained by: module galacticus_error

subroutine: galacticus_signal_handler_sigint
Description: Handle SIGINT signals, by flushing all data and then aborting.
Code lines: 18
Contained by: module galacticus_error

subroutine: galacticus_signal_handler_signsegv
Description: Handle SIGSEGV signals, by flushing all data and then aborting.
Code lines: 18
Contained by: module galacticus_error

file: galacticus.input.path.F90
Description: Contains a module which provides the path for GALACTICUS inputs and scripts.
Code lines: 80

module: galacticus_input_paths
Description: Provides the path for GALACTICUS inputs and scripts.
Code lines: 60
Contained by: file galacticus.input.path.F90
Modules used: iso_varying_string
Used by: function hopkins2007constructor subroutine atomic_data_initialize subroutine chemical_state_atomic_cie_-cloudy_create
19. Source Code Documentation

subroutine cooling_function_atomic_cie_cloudy_create
subroutine galacticus_output_analysis_mass_functions
subroutine galacticus_build_output
subroutine geometry_survey_window_functions_caputi_2011_ukidss_uds
subroutine geometry_survey_window_functions_li_white_2009_sdss
subroutine geometry_survey_window_functions_martin_2010_alfalfa
subroutine filter_response_load
function recfastconstructor
subroutine radiation_igb_file_initialize
function imf_energy_input_rate_noninstantaneous
function imf_metal_yield_rate_noninstantaneous
function imf_recycling_rate_noninstantaneous
subroutine stellar_astrophysics_file_initialize
subroutine supernovae_type_ia_nagashima_initialize
subroutine stellar_tracks_initialize_file
function stellar_population_luminosity
subroutine stellar_population_spectra_conroy_initialize_imf
subroutine critical_overdensity_mass_scaling_wdm_initialize
function power_spectrum_nonlinear_cosmicemu
program test_zhao2009_flat
program test_zhao2009_dark_energy
program test_zhao2009_open
subroutine code_memory_usage

function: galacticus_input_path
Description: Returns the path that GALACTICUS should use as the root for all input data reads.
Code lines: 28
Contained by: module galacticus_input_paths

subroutine: get_path
Description: Retrieve the GALACTICUS input data path from the environment.
Code lines: 13
Contained by: module galacticus_input_paths

file: galacticus.meta.compute_times.F90
Description: Contains a module which implements calculations of the time taken to process merger trees.
Code lines: 109

module: galacticus_meta_compute_times
Description: Implements calculations of the time taken to process merger trees.
Code lines: 89
Contained by: file galacticus.meta.compute_times.F90
Modules used: iso_varying_string
19.1. Program units

Used by: function `merger_tree_construct_mass_-`    program `optimal_sampling_smf`
          `function_sampling_stellar_mf`

function: `galacticus_time_per_tree`
Description: Returns the time (in seconds) to compute a tree of mass `treeRootMass`.
Code lines: 11
Contained by: module `galacticus_meta_compute_times`

subroutine: `galacticus_time_per_tree_initialize`
Description: Initialize the time per tree module.
Code lines: 51
Contained by: module `galacticus_meta_compute_times`
Modules used: `galacticus_error` `galacticus_meta_compute_times_file`
          `input_parameters`

file: `galacticus.meta.compute_times.file.F90`
Description: Contains a module which provides estimates of the time taken to evolve a merger tree in `GALACTICUS`.
Code lines: 92

module: `galacticus_meta_compute_times_file`
Description: Provides estimates of the time taken to evolve a merger tree in `GALACTICUS`.
Code lines: 72
Contained by: file `galacticus.meta.compute_times.file.F90`
Used by: subroutine `galacticus_time_per_tree.initialize`

function: `galacticus_time_per_tree_file`
Description: Provides estimates of the time taken to evolve a merger tree in `GALACTICUS`.
Code lines: 17
Contained by: module `galacticus_meta_compute_times_file`

subroutine: `galacticus_time_per_tree_file_initialize`
Description: Initializes the “file” time per tree module.
Code lines: 38
Contained by: module `galacticus_meta_compute_times_file`
Modules used: `fox_dom` `galacticus_error`
          `input_parameters` `io_xml`
          `iso_varying_string`

file: `galacticus.meta.evolver_profiler.F90`
Description: Contains a module which constructs a profile of `GALACTICUS` ODE evolver statistics.
Code lines: 159

module: `galacticus_meta_evolver_profiler`
Description: Constructs a profile of `GALACTICUS` ODE evolver statistics.
Code lines: 139
Contained by: file `galacticus.meta.evolver_profiler.F90`
Modules used: `hashes`
19. Source Code Documentation

**Used by:** subroutine `galacticus_output_close_file`

**subroutine:** `galacticus_meta_evolver_profile`
- **Description:** Record profiling information on the ODE evolver.
- **Code lines:** 75
- **Contained by:** module `galacticus_meta_evolver_profiler`
- **Modules used:**
  - `arrays_search`
  - `iso_c_binding`
  - `memory_management`
  - `input_parameters`
  - `iso_varying_string`
  - `numerical_ranges`

**subroutine:** `galacticus_meta_evolver_profiler_output`
- **Description:** Outputs collected meta-data on tree evolution.
- **Code lines:** 38
- **Contained by:** module `galacticus_meta_evolver_profiler`
- **Modules used:**
  - `galacticus_hdf5`
  - `iso_varying_string`
  - `numerical_constants_astronomical`

**file:** `galacticus.meta.tree_timing.F90`
- **Description:** Contains a module which records and outputs timing data for processing trees.
- **Code lines:** 204

**module:** `galacticus_meta_tree_timing`
- **Description:** Records and outputs timing data for processing trees.
- **Code lines:** 184
- **Contained by:** file `galacticus.meta.tree_timing.F90`
- **Used by:**
  - subroutine `galacticus_output_close_file`
  - function `galacticus_task_evolve_tree`
  - subroutine `get_tree`

**subroutine:** `meta_tree_timing_initialize`
- **Description:** Initialize the tree timing meta-data module.
- **Code lines:** 27
- **Contained by:** module `galacticus_meta_tree_timing`
- **Modules used:** `input_parameters`

**subroutine:** `meta_tree_timing_output`
- **Description:** Outputs collected meta-data on tree evolution.
- **Code lines:** 35
- **Contained by:** module `galacticus_meta_tree_timing`
- **Modules used:**
  - `galacticus_hdf5`
  - `numerical_constants_astronomical`

**subroutine:** `meta_tree_timing_post_evolve`
- **Description:** Record the CPU time after evolving a tree.
- **Code lines:** 41
- **Contained by:** module `galacticus_meta_tree_timing`
- **Modules used:** `memory_management`

**subroutine:** `meta_tree_timing_pre_construction`
- **Description:** Record the CPU time prior to construction of a tree.
- **Code lines:** 15
19.1. Program units

Contained by: module galacticus_meta_tree_timing

subroutine: meta_tree_timing_pre_evolve
Description: Record the CPU time prior to evolving thisTree.
Code lines: 21
Contained by: module galacticus_meta_tree_timing
Modules used: galacticus_nodes

file: galacticus.output.HDF5.F90
Description: Contains a module which manages HDF5 output from GALACTICUS.
Code lines: 48

module: galacticus_hdf5
Description: Manages HDF5 output from GALACTICUS.
Code lines: 28
Contained by: file galacticus.output.HDF5.F90
Modules used: hdf5
Used by: program galacticus

subroutine: meta_tree_timing_output
subroutine: galacticus_output_analysis_mass_dpndnt_sz_dstrbtins_output
subroutine: galacticus_build_output
subroutine: galacticus_extra_output_halo_fourier_profile
subroutine: star_forma_histo_output_metallcity_split
function: intergalactic_medium_state_internal_update
subroutine: merger_tree_history_write
subroutine: merger_tree_mass_accretion_history_output
subroutine: node_component_black_hole_standard_output merger
subroutine: node_component_dynamics_statistics_bars_output

file: galacticus.output.HDF5.open.F90
Description: Contains a module which handles opening of the GALACTICUS output file.
Code lines: 258

module: galacticus_output_open
Description: Handles opening of the GALACTICUS output file.
Code lines: 238
Contained by: file galacticus.output.HDF5.open.F90
Modules used: galacticus_hdf5
Used by: subroutine: galacticus_merger_tree_output

subroutine: galacticus_meta_evolver_profiler_output
module: galacticus_output_open

function: intergalactic_medium_state_internal_update
subroutine: merger_tree_conditiona_mass_function_output
subroutine: merger_tree_record_evolution_output
subroutine: merger_tree_structure_output
subroutine: node_component_black_hole_standard_output_properties
function: radiation_intergalactic_background_internal_update

1115
function galacticus_task_start

subroutine: galacticus_output_close_file
Description: Close the GALACTICUS output file.
Code lines: 64
Contained by: module galacticus_output_open
Modules used: galacticus_meta_evolver_profiler, galacticus_meta_tree_timing, galacticus_output_analyses_mass_dependent_size_distributions, galacticus_output_analyses_mass_functions, input_parameters, merger_tree_mass_accretion_history, merger_tree_output_structure, merger_trees_conditional_mass_function, merger_trees_dump_evolution, system_command

subroutine: galacticus_output_open_file
Description: Open the file for GALACTICUS output.
Code lines: 157
Contained by: module galacticus_output_open
Modules used: galacticus_build, galacticus_output_halo_models, galacticus_versioning, input_parameters

file: galacticus.output.analysis.cosmology_scalings.F90
Description: Contains a module which computes scaling factors to convert between model and observed cosmologies.
Code lines: 99

module: galacticus_output_analyses_cosmology_scalings
Description: Computes scaling factors to convert between model and observed cosmologies.
Code lines: 79
Contained by: file galacticus.output.analysis.cosmology_scalings.F90
Used by: subroutine galacticus_output_analysis_mass_dependent_size_distributions, subroutine galacticus_output_analysis_mass_functions

subroutine: cosmology_conversion_factors
Description: Compute conversion factors for mass, size, and volume to adjust model galaxy properties to the assumed cosmology of the observations.
Code lines: 70
Contained by: module galacticus_output_analyses_cosmology_scalings
Modules used: cosmology_functions, galacticus_error

file: galacticus.output.analysis.mass_dependent_size_distribution.F90
Description: Contains a module which performs analysis to compute a variety of mass-dependent size functions.
Code lines: 558

module: galacticus_output_analyses_mass_dependent_size_distributions
Description: Performs analysis to compute a variety of mass-dependent size functions. Currently supported mass functions include:

- The SDSS late-type galaxy size distributions from Shen et al. [2003].
19. Source Code Documentation

**Code lines:** 538  
**Contained by:** file `galacticus.output.analysis.mass_dependent_size_distribution.F90`  
**Modules used:**  
- `galactic_structure_options`  
- `galacticus_nodes`  
- `iso_c_binding`  
- `numerical_constants_prefixes`  
**Used by:** subroutine `galacticus_output_close_file`  
subroutine `galacticus_merger_tree_output`  

**subroutine:** `galacticus_output_analysis_mass_dpndnt_sz_dstrbtins`  
**Description:** Construct a mass functions to compare to various observational determinations.  
**Code lines:** 323  
**Contained by:** module `galacticus_output_analyses_mass_dpndnt_sz_dstrbtins`  
**Modules used:**  
- `cosmology_functions`  
- `cosmology_parameters`  
- `dark_matter_halo_scales`  
- `fox_dom`  
- `galacticus_error`  
- `galacticus_input_paths`  
- `galacticus_nodes`  
- `galacticus_output_analyses_mass_dpndnt_sz_dstrbtins`  
- `galacticus_output_times`  
- `cosmology_scalings`  
- `input_parameters`  
- `iso_c_binding`  
- `io_xml`  
- `iso_varying_string`  
- `memory_management`  
- `numerical_comparison`  
- `string_handling`  
- `vectors`  

**subroutine:** `galacticus_output_analysis_mass_dpndnt_sz_dstrbtins_output`  
**Description:** Outputs SDSS $z \approx 0.07$ stellar mass function to file.  
**Code lines:** 67  
**Contained by:** module `galacticus_output_analyses_mass_dpndnt_sz_dstrbtins`  
**Modules used:** `galacticus_hdf5`  

**function:** `map_radius_sdss_size_function_z0_07`  
**Description:** Maps scale radii into Petrosian $r_{50}$ radii for the SDSS disk size analysis. Also converts from Mpc to kpc.  
**Code lines:** 9  
**Contained by:** module `galacticus_output_analyses_mass_dpndnt_sz_dstrbtins`  

**type:** `sizefunction`  
**Code lines:** 18  
**Contained by:** module `galacticus_output_analyses_mass_dpndnt_sz_dstrbtins`  

**type:** `sizefunctiondescriptor`  
**Code lines:** 14  
**Contained by:** module `galacticus_output_analyses_mass_dpndnt_sz_dstrbtins`  

**type:** `sizefunctionwork`  
**Code lines:** 4  
**Contained by:** module `galacticus_output_analyses_mass_dpndnt_sz_dstrbtins`  

**file:** `galacticus.output.analysis.mass_functions.F90`  
**Description:** Contains a module which performs analysis to compute a variety of mass functions.
19.1. Program units

module: galacticus_output_analyses_mass_functions
Description: Performs analysis to compute a variety of mass functions. Currently supported mass functions include:

- The $z \approx 0.07$ stellar mass function in the SDSS measured by Li and White [2009].
  Assumes 0.07 dex random errors on stellar masses. This is approximate, but motivated by the discussion of Behroozi et al. [2010].

Code lines: 705
Contained by: file galacticus.output.analysis.mass_functions.F90
Modules used: galactic_structure_options galacticus_nodes iso_c_binding
Used by: subroutine galacticus_output_close_file subroutine galacticus_merger_tree_output

subroutine: alfalfa_hi_mass_function_z0_00_initialize
Description: Initializes the ALFALFA HI mass function calculation by reading in required parameters.
Code lines: 685
Contained by: module galacticus_output_analyses_mass_functions
Modules used: galacticus_nodes

subroutine: galacticus_output_analysis_mass_functions
Description: Construct a mass functions to compare to various observational determinations.
Code lines: 327
Contained by: module galacticus_output_analyses_mass_functions
Modules used: cosmology_functions cosmology_parameters galactic_structure_enclosed_masses galacticus_error galacticus_input_paths galacticus_nodes galacticus_output_analyses_cosmology_scalings galacticus_output_analyses_times input_parameters io_hdf5 iso_c_binding iso_varying_string memory_management numerical_comparison string_handling

subroutine: galacticus_output_analysis_mass_functions_output
Description: Outputs SDSS $z \approx 0.07$ stellar mass function to file.
Code lines: 61
Contained by: module galacticus_output_analyses_mass_functions
Modules used: galacticus_hdf5 numerical_constants_astronomical

subroutine: load_primus_mass_function
Description: Load the specified mass function from the PRIMUS stellar mass function dataset.
Code lines: 81
Contained by: module galacticus_output_analyses_mass_functions
Modules used: cosmology_functions cosmology_parameters fox_dom galacticus_error galacticus_input_paths io_xml
function: map_mass_alfalfa_hi_mass_function_z0_00
Description: Maps gas masses into HI masses for the ALFALFA survey analysis. Assumes a constant gas to HI mass conversion factor of 0.54 [Power et al., 2010].
Code lines: 16
Contained by: module galacticus_output_analyses_mass_functions
Modules used: numerical_constants_astronomical

function: mass_error_alfalfa_hi_mass_function_z0_00
Description: Computes errors on $\log_{10}(\text{HI masses})$ for the ALFALFA survey analysis. Uses a simple fitting function. See constraints/dataAnalysis/hiMassFunction_ALFALFA_-z0.00/alfalfaHIMassErrorModel.pl for details.
Code lines: 15
Contained by: module galacticus_output_analyses_mass_functions

type: massfunction
Code lines: 17
Contained by: module galacticus_output_analyses_mass_functions

type: massfunctiondescriptor
Code lines: 11
Contained by: module galacticus_output_analyses_mass_functions

type: massfunctionwork
Code lines: 3
Contained by: module galacticus_output_analyses_mass_functions

file: galacticus.output.build.F90
Description: Contains a module which implements writing of GALACTICUS build information to the GALACTICUS output file.
Code lines: 212

module: galacticus_build
Description: Implements writing of GALACTICUS build information to the GALACTICUS output file.
Code lines: 189
Contained by: file galacticus.output.build.F90
Modules used: iso_varying_string
Used by: subroutine galacticus_output_open_file module input_parameters

subroutine: galacticus_build_output
Description: Output build information to the main output file.
Code lines: 94
Contained by: module galacticus_build
Modules used: iso_varying_string file_utilities
                      fox_common galacticus_error
galacticus_hdf5 galacticus_input_paths
iso_varying_string string_handling
**function: galacticus_build_string**

*Description:* Returns a string describing the build environment of GALACTICUS.

*Code lines:* 71

*Contained by:* module galacticus_build

*Modules used:* fox_common
galacticus_error
galacticus_hdf5
iso_varying_string
string_handling

**file: galacticus.output.merger_tree.F90**

*Description:* Contains a module which implements writing a merger tree to the GALACTICUS output file.

*Code lines:* 857

**module: galacticus_output_merger_tree**

*Description:* Implements writing a merger tree to the GALACTICUS output file.

*Code lines:* 837

*Contained by:* file galacticus.output.merger_tree.F90

*Modules used:* iso_c_binding
iso_varying_string
kind_numbers

*Used by:* function galacticus_task_end

*Function:* galacticus_task_evolve_tree

**subroutine: allocate_buffers**

*Description:* Allocate buffers for storage of properties.

*Code lines:* 35

*Contained by:* module galacticus_output_merger_tree

*Modules used:* accretion_halos
cooling_radii
galacticus_merger_tree_output_-_filter_lightcones
galacticus_output_halo_models
galacticus_output_tree_half_light_-_properties
galacticus_output_tree_mass_profiles
galacticus_output_trees_descendents
galacticus_output_trees_links
galacticus_output_trees_redshifts
galacticus_output_trees_satellite_-_extremum
galacticus_output_trees_satellite_-_host

galacticus_output_trees_main_branch
galacticus_output_trees_rotation_curve
galacticus_output_trees_satellite_host

galacticus_output_tree_half_mass_radii
galacticus_output_trees_density_contrasts
galacticus_output_trees_final_descendants

galacticus_output_most_massive_progenitors
galacticus_output_trees_final_descendants

**subroutine: count_properties**

*Description:* Count up the number of properties that will be output.

*Code lines:* 84

*Contained by:* module galacticus_output_merger_tree

*Modules used:* cooling_cold_mode_infall_rates
cooling_rates
galacticus_nodes
galacticus_output_halo_models
galacticus_output_tree_half_mass_radii
galacticus_output_trees_density_contrasts
galacticus_output_trees_final_descendants

galacticus_output_trees_satellite_host

19. Source Code Documentation

```
galacticus_output_trees_satellite_status
galacticus_output_trees_velocity_dispersion
node_component_black_hole_simple
node_component_merging_statistics_recent

subroutine: double_buffer_dump
  Description: Dump the contents of the double precision properties buffer to the GALACTICUS output file.
  Code lines: 21
  Contained by: module galacticus_output_merger_tree
  Modules used: iso_c_binding

subroutine: double_buffer_extend
  Description: Extend the size of the double buffer.
  Code lines: 14
  Contained by: module galacticus_output_merger_tree
  Modules used: memory_management

subroutine: establish_property_names
  Description: Set names for the properties.
  Code lines: 86
  Contained by: module galacticus_output_merger_tree
  Modules used: accretion_halos
cooling_radii
galacticus_merge_tree_output_filter_lightcones
galacticus_output_halo_models
galacticus_output_tree_half_light_properties
galacticus_output_tree_mass_profiles
galacticus_output_trees_descendents
galacticus_output_trees_links
galacticus_output_trees_redshifts
galacticus_output_trees_satellite_extremum
galacticus_output_trees_satellite_status
galacticus_output_trees_velocity_dispersion
node_component_black_hole_simple
node_component_merging_statistics_recent

cooling_cold_mode_infall_rates
cooling_rates
galacticus_nodes
galacticus_output_most_massive_progenitors
galacticus_output_tree_half_mass_radii
galacticus_output_trees_density_contrasts
galacticus_output_trees_final_descendants
galacticus_output_trees_main_branch
galacticus_output_trees_rotation_curve
galacticus_output_trees_satellite_host
galacticus_output_trees_satellite_indices
galacticus_output_trees_virial
node_component_black_hole_standard
```
19.1. Program units

**subroutine: galacticus_merger_tree_output**

*Description:* Write properties of nodes in thisTree to the GALACTICUS output file.

*Code lines:* 332

*Contained by:* module galacticus_output_merger_tree

*Modules used:*
- accretion_halos
- cooling_radii
- galactic_structure_radii
- galacticus_merger_tree_output_filters
- galacticus_output_analyses_mass_density_contrasts
- galacticus_output_halo_models
- galacticus_output_open
- galacticus_output_tree_half_mass_radii
- galacticus_output_trees_density_contrasts
- galacticus_output_trees_final_descendants
- galacticus_output_trees_main_branch
- galacticus_output_trees_rotation_curve
- galacticus_output_trees_satellite_host
- galacticus_output_trees_tree_indices
- galacticus_output_trees_virial
- iso_c_binding
- node_component_black_hole_simple
- node_component_disk_exponential
- node_component_inter_output_standard
- node_component_merging_statistics_recent
- cooling_cold_mode_infall_rates
- cooling_rates
- galacticus_merger_tree_output_filter_lightcones
- galacticus_nodes
- galacticus_output_analyses_mass_functions
- galacticus_output_most_massive_progenitors
- galacticus_output_tree_half_light_properties
- galacticus_output_tree_mass_profiles
- galacticus_output_trees_descendents
- galacticus_output_trees_links
- galacticus_output_trees_redshifts
- galacticus_output_trees_satellite_extremum
- galacticus_output_trees_satellite_status
- galacticus_output_trees_velocity_dispersion
- input_parameters
- merger_tree_timesteps_record_evolution
- node_component_black_hole_standard
- node_component_dynamics_statistics_bars
- node_component_mass_flow_statistics_standard
- node_component_spheroid_standard

**subroutine: galacticus_merger_tree_output_finalize**

*Description:* Finalize merger tree output by closing any open groups.

*Code lines:* 26

*Contained by:* module galacticus_output_merger_tree

**subroutine: galacticus_merger_tree_output_make_group**

*Description:* Make an group in the GALACTICUS file in which to store thisTree.

*Code lines:* 26
19. Source Code Documentation

```
1. Source Code Documentation

**Contained by:** module galacticus_output_merger_tree

**Modules used:**
- galacticus_nodes
- iso_c_binding
- numerical_constants_astronomical
- string_handling

**subroutine:** integer_buffer_dump

**Description:** Dump the contents of the integer properties buffer to the GALACTICUS output file.

**Code lines:** 21

**Contained by:** module galacticus_output_merger_tree

**Modules used:** iso_c_binding

**subroutine:** integer_buffer_extend

**Description:** Extend the size of the integer buffer.

**Code lines:** 14

**Contained by:** module galacticus_output_merger_tree

**Modules used:** memory_management

**subroutine:** make_output_group

**Description:** Create a group in which to store this output.

**Code lines:** 102

**Contained by:** module galacticus_output_merger_tree

**Modules used:**
- cosmology_functions
- galacticus_output_halo_models
- iso_c_binding
- memory_management
- numerical_constants_astronomical
- string_handling

**type:** outputgroup

**Description:** Type used for output group information.

**Code lines:** 6

**Contained by:** module galacticus_output_merger_tree

**file:** galacticus.output.merger_tree.density_contrasts.F90

**Description:** Contains a module which handles outputting of node density contrast properties (radii and masses).

**Code lines:** 265

**module:** galacticus_output_trees_density_contrasts

**Description:** Handles outputting of node density contrast properties (radii and masses).

**Code lines:** 245

**Contained by:** file galacticus.output.merger_tree.density_contrasts.F90

**Modules used:**
- dark_matter_halo_scales
- galactic_structure_enclosed_masses
- galactic_structure_options
- kind_numbers

**subroutine:** galacticus_output_tree_density_contrast

**Description:** Store density contrast properties in the GALACTICUS output file buffers.

**Code lines:** 49

**Contained by:** module galacticus_output_trees_density_contrasts

**Modules used:**
- dark_matter_halo_scales
- galactic_structure_enclosed_masses
- galactic_structure_options
- kind_numbers
```

1124
root_finder

subroutine: galacticus_output_tree_density_contrast_initialize
Description: Initializes the module by determining whether or not density contrast data should be output and getting a list of overdensities.
Code lines: 83
Contained by: module galacticus_output_trees_density_contrasts
Modules used: cosmology_parameters, galactic_structure_options, input_parameters, memory_management

subroutine: galacticus_output_tree_density_contrast_names
Description: Set the names of density contrast properties to be written to the GALACTICUS output file.
Code lines: 28
Contained by: module galacticus_output_trees_density_contrasts
Modules used: numerical_constants_astronomical

subroutine: galacticus_output_tree_density_contrast_property_count
Description: Account for the number of density contrast properties to be written to the GALACTICUS output file.
Code lines: 13
Contained by: module galacticus_output_trees_density_contrasts

function: mean_density_contrast_root
Description: Root function used in finding the radius that encloses a given density contrast.
Code lines: 17
Contained by: module galacticus_output_trees_density_contrasts
Modules used: cosmology_functions, galactic_structure_enclosed_masses, galactic_structure_options, numerical_constants_math

file: galacticus.output.merger_tree.descendents.F90
Description: Contains a module which handles outputting of tree descendent data to the GALACTICUS output file.
Code lines: 217

module: galacticus_output_trees_descendents
Description: Handles outputting of tree descendent data to the GALACTICUS output file.
Code lines: 197
Contained by: file galacticus.output.merger_tree.descendents.F90
 Modules used: galacticus_nodes
 Used by: subroutine count_properties, subroutine establish_property_names, subroutine galacticus_merger_tree_output

subroutine: galacticus_output_tree_descendents
Description: Store descendent properties in the GALACTICUS output file buffers.
Code lines: 88
Contained by: module galacticus_output_trees_descendents
Modules used: galacticus_output_times, kind_numbers
subroutine: galacticus_output_tree_descendants_initialize
Description: Initializes the module by determining whether or not descendent data should be output.
Code lines: 30
Contained by: module galacticus_output_trees_descendents
Modules used: galacticus_error
iso_varying_string
input_parameters

subroutine: galacticus_output_tree_descendants_names
Description: Set the names of descendent properties to be written to the GALACTICUS output file.
Code lines: 28
Contained by: module galacticus_output_trees_descendents

subroutine: galacticus_output_tree_descendants_property_count
Description: Account for the number of descendent properties to be written to the GALACTICUS output file.
Code lines: 13
Contained by: module galacticus_output_trees_descendents

file: galacticus.output.merger_tree.filters.F90
Description: Contains a module which provides filtering of output.
Code lines: 148

module: galacticus_merger_tree_output_filters
Description: Provides filtering of output.
Code lines: 128
Contained by: file galacticus.output.merger_tree.filters.F90
Modules used: iso_varying_string
galacticus_merger_tree_output_filter_lightcones
galacticus_merger_tree_output_filter_luminosities
galacticus_merger_tree_output_filter_stellar_masses
galacticus_merger_tree_output_stllr_mss_mrpflgs
galacticus_nodes

function: galacticus_merger_tree_output_filter
Description: Return true if thisNode should be included in the output. Always arbitrary filters to block output of thisNode.
Code lines: 43
Contained by: module galacticus_merger_tree_output_filters

subroutine: galacticus_merger_tree_output_filter_initialize
Description: Initialize the output filter subsystem.
Code lines: 63
Contained by: module galacticus_merger_tree_output_filters
Modules used: galacticus_merger_tree_output_filter_lightcones
galacticus_merger_tree_output_filter_luminosities
19.1. Program units

file: galacticus.output.merger_tree.filters.lightcone.F90
Description: Contains a module which filters output for lightcone geometry.
Code lines: 567

module: galacticus_merger_tree_output_filter_lightcones
Description: Filters output for lightcone geometry.
Code lines: 547
Contained by: file galacticus.output.merger_tree.filters.lightcone.F90
Used by: subroutine count_properties subroutine establish_property_names subroutine galacticus_merger_tree_output output subroutine galacticus_merger_tree_output output_filter subroutine galacticus_merger_tree_output output_filter_initialize

function: filter_lightcone_get_coordinates
Description: Extract a vector of coordinates from an XML DOM element.
Code lines: 12
Contained by: module galacticus_merger_tree_output_filter_lightcones
Modules used: fox.dom galacticus_error

subroutine: galacticus_merger_tree_lightcone_geometry_initialize
Description: Initialize geometry for lightcone output.
Code lines: 152
Contained by: module galacticus_merger_tree_output_filter_lightcones
Modules used: cosmology_functions cosmology_parameters fox.dom galacticus_error input_parameters io.xml memory_management numerical_constants_astronomical trigonometric_functions vectors

function: inversecosineintegral
Description: Integral of \( \sin(x) \cdot \cos^{-1}(a/tan(x)) \) evaluated using Wolfram Alpha.
Code lines: 10
Contained by: subroutine galacticus_merger_tree_lightcone_geometry_initialize

subroutine: galacticus_merger_tree_output_filter_lightcone
Description: Determines whether thisNode lies within a lightcone and, therefore, should be output.
Code lines: 82
Contained by: module galacticus_merger_tree_output_filter_lightcones
Modules used: arrays_search cosmology_functions galacticus_nodes iso_c_binding vectors

1127
subroutine: galacticus_merger_tree_output_filter_lightcone_initialize
Description: Initializes the lightcone filter module.
Code lines: 16
Contained by: module galacticus_merger_tree_output_filter_lightcones
Modules used: iso_varying_string

subroutine: galacticus_output_tree_lightcone
Description: Store link properties in the GALACTICUS output file buffers.
Code lines: 22
Contained by: module galacticus_merger_tree_output_filter_lightcones
Modules used: galacticus_nodes kind_numbers

subroutine: galacticus_output_tree_lightcone_names
Description: Set the names of link properties to be written to the GALACTICUS output file.
Code lines: 126
Contained by: module galacticus_merger_tree_output_filter_lightcones
Modules used: galacticus_nodes numerical_constants_astronomical

subroutine: galacticus_output_tree_lightcone_property_count
Description: Account for the number of link properties to be written to the GALACTICUS output file.
Code lines: 10
Contained by: module galacticus_merger_tree_output_filter_lightcones
Modules used: galacticus_nodes

subroutine: merger_tree_prune_lightcone
Description: Prune branches from thisTree.
Code lines: 57
Contained by: module galacticus_merger_tree_output_filter_lightcones
Modules used: galacticus_nodes input_parameters

file: galacticus.output.merger_tree.filters.luminosities.F90
Description: Contains a module which filters output on stellar mass.
Code lines: 144

module: galacticus_merger_tree_output_filter_luminosities
Description: Filters output for lightcone geometry.
Code lines: 124
Contained by: file galacticus.output.merger_tree.filters.luminosities.F90
Used by: function galacticus_merger_tree_-output_filter subroutine galacticus_merger_tree_-output_filter_initialize

subroutine: galacticus_merger_tree_output_filter_luminosity
Description: Determines whether thisNode has sufficient stellar mass to be output.
Code lines: 49
Contained by: module galacticus_merger_tree_output_filter_luminosities
Modules used: galactic_structure_enclosed_masses galactic_structure_options galacticus_nodes stellar_luminosities_structure

1128
subroutine: galacticus_merger_tree_output_filter_luminosity_initialize
Description: Initializes the stellar mass filter module.
Code lines: 49
Contained by: module galacticus_merger_tree_output_filter_luminosities
Modules used: galacticus_error input_parameters
iso_varying_string memory_management
stellar_luminosities_structure

file: galacticus.output.merger_tree.filters.stellar_mass.F90
Description: Contains a module which filters output on stellar mass.
Code lines: 94

module: galacticus_merger_tree_output_filter_stellar_masses
Description: Filters output for lightcone geometry.
Code lines: 74
Contained by: file galacticus.output.merger_tree.filters.stellar_mass.F90
Used by: function galacticus_merger_tree_output_filter subroutine galacticus_merger_tree_output_filter_initialize

subroutine: galacticus_merger_tree_output_filter_stellar_mass
Description: Determines whether thisNode has sufficient stellar mass to be output.
Code lines: 19
Contained by: module galacticus_merger_tree_output_filter_stellar_masses
Modules used: galactic_structure_enclosed_masses galactic_structure_options
galacticus_nodes

subroutine: galacticus_merger_tree_output_filter_stellar_mass_initialize
Description: Initializes the stellar mass filter module.
Code lines: 30
Contained by: module galacticus_merger_tree_output_filter_stellar_masses
Modules used: input_parameters iso_varying_string

file: galacticus.output.merger_tree.filters.stellar_mass_morphology.F90
Description: Contains a module which filters output on stellar mass spheroid-to-total ratio.
Code lines: 111

module: galacticus_merger_tree_output_stllr_mss_mrphlgs
Description: Filters output on stellar mass spheroid-to-total ratio.
Code lines: 91
Contained by: file galacticus.output.merger_tree.filters.stellar_mass_morphology.F90
Used by: function galacticus_merger_tree_output_filter subroutine galacticus_merger_tree_output_filter_initialize

subroutine: galacticus_merger_tree_output_stllr_mss_mrphlgy
Description: Determines whether thisNode has sufficient stellar mass to be output.
Code lines: 25
Contained by: module galacticus_merger_tree_output_stllr_mss_mrphlgs
19. Source Code Documentation

**Modules used:** galacticus_nodes

**subroutine:** galacticus_merger_tree_output_stllr_mss_mrphlgy_initialize

*Description:* Initializes the stellar mass filter module.
*Code lines:* 41
*Contained by:* module galacticus_merger_tree_output_stllr_mss_mrphlgys

**Modules used:** input_parameters iso_varying_string

**file:** galacticus.output.merger_tree.final_descendent.F90

*Description:* Contains a module which handles outputting of tree final descendent data to the GALACTICUS output file.
*Code lines:* 165

**module:** galacticus_output_trees_final_descendents

*Description:* Handles outputting of tree final descendent data to the GALACTICUS output file.
*Code lines:* 145
*Contained by:* file galacticus.output.merger_tree.final_descendent.F90

**Modules used:** galacticus_nodes

**Used by:** subroutine count_properties subroutine establish_property_names subroutine galacticus_merger_tree_output

**subroutine:** galacticus_output_tree_final_descendents

*Description:* Store descendent properties in the GALACTICUS output file buffers.
*Code lines:* 35
*Contained by:* module galacticus_output_trees_final_descendents

**Modules used:** galacticus_nodes galacticus_output_times kind_numbers

**subroutine:** galacticus_output_tree_final_descendents_initialize

*Description:* Initializes the module by determining whether or not descendent data should be output.
*Code lines:* 30
*Contained by:* module galacticus_output_trees_final_descendents

**Modules used:** galacticus_error input_parameters iso_varying_string

**subroutine:** galacticus_output_tree_final_descendents_names

*Description:* Set the names of descendent properties to be written to the GALACTICUS output file.
*Code lines:* 28
*Contained by:* module galacticus_output_trees_final_descendents

**subroutine:** galacticus_output_tree_final_descendents_property_count

*Description:* Account for the number of descendent properties to be written to the GALACTICUS output file.
*Code lines:* 14
*Contained by:* module galacticus_output_trees_final_descendents

**Modules used:** galacticus_nodes

**file:** galacticus.output.merger_tree.half_light_properties.F90
19.1. Program units

**Description:** Contains a module which handles outputting of galaxy half-light properties (radii and masses).

**Code lines:** 196

**module:** galacticus_output_tree_half_light_properties

**Description:** Handles outputting of galaxy half-light radii and associated masses.

**Code lines:** 176

**Contained by:** file galacticus.output.merger_tree.half_light_properties.F90

**Modules used:** galacticus_nodes

**Used by:** subroutine count_properties subroutine establish_property_names subroutine galacticus_merger_tree_output

**subroutine:** galacticus_output_tree_half_light

**Description:** Store density contrast properties in the GALACTICUS output file buffers.

**Code lines:** 45

**Contained by:** module galacticus_output_tree_half_light_properties

**Modules used:** galactic_structure_enclosed_masses galactic_structure_options kind_numbers stellar_luminosities_structure

**subroutine:** galacticus_output_tree_half_light_initialize

**Description:** Initializes the module by determining whether or not half-light radius data should be output.

**Code lines:** 31

**Contained by:** module galacticus_output_tree_half_light_properties

**Modules used:** input_parameters galacticus_output_tree_half_light_properties stellar_luminosities_structure

**subroutine:** galacticus_output_tree_half_light_names

**Description:** Set the names of half-light properties to be written to the GALACTICUS output file.

**Code lines:** 48

**Contained by:** module galacticus_output_tree_half_light_properties

**Modules used:** iso_varying_string numerical_constants_astronomical stellar_luminosities_structure

**subroutine:** galacticus_output_tree_half_light_property_count

**Description:** Account for the number of half-light properties to be written to the GALACTICUS output file.

**Code lines:** 14

**Contained by:** module galacticus_output_tree_half_light_properties

**Modules used:** stellar_luminosities_structure

**file:** galacticus.output.merger_tree.half_mass_radius.F90

**Description:** Contains a module which handles outputting of galaxy half-mass radii.

**Code lines:** 153

**module:** galacticus_output_tree_half_mass_radii

**Description:** Handles outputting of galaxy half-mass radii.

**Code lines:** 133

**Contained by:** file galacticus.output.merger_tree.half_mass_radius.F90

**Modules used:** galacticus_nodes
19. Source Code Documentation

Used by: subroutine count_properties subroutine establish_property_names subroutine galacticus_merger_tree_-output

subroutine: galacticus_output_tree_half_mass
Description: Store density contrast properties in the GALACTICUS output file buffers.
Code lines: 26
Contained by: module galacticus_output_tree_half_mass_radii
Modules used: galactic_structure_enclosed_masses galactic_structure_options kind_numbers stellar_luminosities_structure

subroutine: galacticus_output_tree_half_mass_initialize
Description: Initializes the module by determining whether or not half-mass radii should be output.
Code lines: 27
Contained by: module galacticus_output_tree_half_mass_radii
Modules used: input_parameters stellar_luminosities_structure

subroutine: galacticus_output_tree_half_mass_names
Description: Set the names of half-light properties to be written to the GALACTICUS output file.
Code lines: 31
Contained by: module galacticus_output_tree_half_mass_radii
Modules used: iso_varying_string numerical_constants_astronomical stellar_luminosities_structure

subroutine: galacticus_output_tree_half_mass_property_count
Description: Account for the number of half-light properties to be written to the GALACTICUS output file.
Code lines: 14
Contained by: module galacticus_output_tree_half_mass_radii
Modules used: stellar_luminosities_structure

file: galacticus.output.merger_tree.halo_model.F90
Description: Contains a module which handles outputting of data required by the halo model of galaxy clustering to the GALACTICUS output file.
Code lines: 406

module: galacticus_output_halo_models
Description: Handles outputting of data required by the halo model of galaxy clustering to the GALACTICUS output file.
Code lines: 386
Contained by: file galacticus.output.merger_tree.halo_model.F90
Used by: subroutine galacticus_output_open_file subroutine count_properties subroutine establish_property_names subroutine galacticus_merger_tree_-output subroutine make_output_group

subroutine: galacticus_extra_output_halo_fourier_profile
Description: Store Fourier-space halo profiles to the output file.
Code lines: 75
19.1. Program units

**module** galacticus_output_halo_models

**Description:**
Output the linear theory power spectrum to the main output file.

**Code lines:** 22

**Contained by:**
module galacticus_output_halo_models

**subroutine:** galacticus_growth_factor_output

**Description:**
Output the linear theory power spectrum to the main output file.

**Code lines:** 78

**Contained by:**
module galacticus_output_halo_models

**subroutine:** galacticus_linear_power_spectrum_output

**Description:**
Output the linear theory power spectrum to the main output file.

**Code lines:** 64

**Contained by:**
module galacticus_output_halo_models

**subroutine:** galacticus_output_halo_model

**Description:**
Store halo model properties in the GALACTICUS output file buffers.

**Code lines:** 41

**Contained by:**
module galacticus_output_halo_models

**subroutine:** galacticus_output_halo_model_initialize

**Description:**
Initializes the module by determining whether or not halo model data should be output.

**Code lines:** 17

**Contained by:**
module galacticus_output_halo_models

**subroutine:** galacticus_output_halo_model_names

**Description:**
Set the names of halo model properties to be written to the GALACTICUS output file.

**Code lines:** 11

**Contained by:**
module galacticus_output_halo_models

**subroutine:** galacticus_output_halo_model_property_count

**Description:**
Account for the number of halo model properties to be written to the GALACTICUS output file.

**Code lines:** 66

**Contained by:**
module galacticus_output_halo_models

**file:** galacticus.output.merger_tree.links.F90

**Description:**
Contains a module which handles outputting of tree link data to the GALACTICUS output file.
19. Source Code Documentation

Code lines: 169

module: galacticus_output_trees_links
Description: Handles outputting of tree link data to the GALACTICUS output file.
Code lines: 149
Contained by: file galacticus.output.merger_tree.links.F90
Modules used: galacticus_nodes
Used by: subroutine count_properties subroutine establish_property_names subroutine galacticus_merger_tree_output

subroutine: galacticus_output_tree_links
Description: Store link properties in the GALACTICUS output file buffers.
Code lines: 28
Contained by: module galacticus_output_trees_links
Modules used: kind_numbers

subroutine: galacticus_output_tree_links_names
Description: Set the names of link properties to be written to the GALACTICUS output file.
Code lines: 82
Contained by: module galacticus_output_trees_links

subroutine: galacticus_output_tree_links_property_count
Description: Account for the number of link properties to be written to the GALACTICUS output file.
Code lines: 9
Contained by: module galacticus_output_trees_links

file: galacticus.output.merger_tree.main_branch.F90
Description: Contains a module which handles outputting of tree main branch data to the GALACTICUS output file.
Code lines: 151

module: galacticus_output_trees_main_branch
Description: Handles outputting of tree main branch data to the GALACTICUS output file.
Code lines: 131
Contained by: file galacticus.output.merger_tree.main_branch.F90
Used by: subroutine count_properties subroutine establish_property_names subroutine galacticus_merger_tree_output

subroutine: galacticus_output_tree_main_branch
Description: Store mainBranch properties in the GALACTICUS output file buffers.
Code lines: 24
Contained by: module galacticus_output_trees_main_branch
Modules used: galacticus_nodes kind_numbers

subroutine: galacticus_output_tree_main_branch_initialize
Description: Initialize the “main branch status” output module.
Code lines: 29
19.1. Program units

Included by: module `galacticus_output_trees_main_branch`
Modules used: `input_parameters`

**subroutine: galacticus_output_tree_main_branch_names**
*Description:* Set the names of main branch properties to be written to the GALACTICUS output file.
*Code lines:* 28
*Included by:* module `galacticus_output_trees_main_branch`
*Modules used:* `galacticus_nodes`

**subroutine: galacticus_output_tree_main_branch_property_count**
*Description:* Account for the number of main branch properties to be written to the GALACTICUS output file.
*Code lines:* 13
*Included by:* module `galacticus_output_trees_main_branch`
*Modules used:* `galacticus_nodes`

**file: galacticus.output.merger_tree.mass_profile.F90**
*Description:* Contains a module which handles outputting of node mass profiles.
*Code lines:* 177

**module: galacticus_output_tree_mass_profiles**
*Description:* Handles outputting of node mass profiles.
*Code lines:* 157
*Included by:* file `galacticus.output.merger_tree.mass_profile.F90`
*Modules used:* `galacticus_nodes`
*Used by:* subroutine `count_properties` subroutine `establish_property_names` subroutine `galacticus_merger_tree_output`

**subroutine: galacticus_output_tree_mass_profile**
*Description:* Store density contrast properties in the GALACTICUS output file buffers.
*Code lines:* 34
*Included by:* module `galacticus_output_tree_mass_profiles`
*Modules used:* `galactic_structure_enclosed_masses` `galactic_structure_options` `kind_numbers`

**subroutine: galacticus_output_tree_mass_profile_initialize**
*Description:* Initializes the module by determining whether or not half-light radius data should be output.
*Code lines:* 45
*Included by:* module `galacticus_output_tree_mass_profiles`
*Modules used:* `input_parameters` `memory_management`

**subroutine: galacticus_output_tree_mass_profile_names**
*Description:* Set the names of half-light properties to be written to the GALACTICUS output file.
*Code lines:* 24
*Included by:* module `galacticus_output_tree_mass_profiles`
*Modules used:* `numerical_constants_astronomical`

**subroutine: galacticus_output_tree_mass_profile_property_count**
19. Source Code Documentation

**Description:** Account for the number of half-light properties to be written to the GALACTICUS output file.

**Code lines:** 13  
**Contained by:** module `galacticus_output_tree_mass_profiles`

**file:** `galacticus.output.merger_tree.most_massive_progenitor.F90`  
**Description:** Contains a module which handles outputting a flag for the most massive progenitor in a tree.

**Code lines:** 174

**module:** `galacticus_output_most_massive_progenitors`  
**Description:** Handles outputting a flag for the most massive progenitor in a tree.

**Code lines:** 154  
**Contained by:** file `galacticus.output.merger_tree.most_massive_progenitor.F90`  
**Used by:** subroutine `count_properties`  
subroutine `establish_property_names`  
subroutine `galacticus_merger_tree_output`  
subroutine `galacticus_output_most_massive_progenitor`

**Description:** Store link properties in the GALACTICUS output file buffers.

**Code lines:** 50  
**Contained by:** module `galacticus_output_most_massive_progenitors`  
**Modules used:** `galacticus_nodes`  
`kind_numbers`

**subroutine:** `galacticus_output_most_massive_progenitor_initialize`  
**Description:** Initialize the module that outputs flags for the most massive progenitor in a tree.

**Code lines:** 26  
**Contained by:** module `galacticus_output_most_massive_progenitors`  
**Modules used:** `input_parameters`

**subroutine:** `galacticus_output_most_massive_progenitor_names`  
**Description:** Set the names of link properties to be written to the GALACTICUS output file.

**Code lines:** 28  
**Contained by:** module `galacticus_output_most_massive_progenitors`  
**Modules used:** `galacticus_nodes`

**subroutine:** `galacticus_output_most_massive_progenitor_property_count`  
**Description:** Account for the number of link properties to be written to the GALACTICUS output file.

**Code lines:** 13  
**Contained by:** module `galacticus_output_most_massive_progenitors`  
**Modules used:** `galacticus_nodes`

**file:** `galacticus.output.merger_tree.redshifts.F90`  
**Description:** Contains a module which handles outputting of node redshifts to the GALACTICUS output file.

**Code lines:** 151

**module:** `galacticus_output_trees_redshifts`  
**Description:** Handles outputting of node redshift data to the GALACTICUS output file.

**Code lines:** 131
19.1. Program units

**Contained by:** file `galacticus.output.merger_tree.redshifts.F90`

**Modules used:** `galacticus_nodes`

**Used by:** subroutine `count_properties` subroutine `establish_property_names` subroutine `galacticus_merger_tree_output`

**subroutine:** `galacticus_output_redshifts`

**Description:** Store link properties in the GALACTICUS output file buffers.

**Code lines:** 21

**Contained by:** module `galacticus_output_trees_redshifts`

**Modules used:** `cosmology_functions` `kind_numbers`

**subroutine:** `galacticus_output_redshifts_initialize`

**Description:** Initialize the “redshift” output module.

**Code lines:** 36

**Contained by:** module `galacticus_output_trees_redshifts`

**Modules used:** `input_parameters`

**subroutine:** `galacticus_output_redshifts_names`

**Description:** Set the names of link properties to be written to the GALACTICUS output file.

**Code lines:** 25

**Contained by:** module `galacticus_output_trees_redshifts`

**subroutine:** `galacticus_output_redshifts_property_count`

**Description:** Account for the number of link properties to be written to the GALACTICUS output file.

**Code lines:** 10

**Contained by:** module `galacticus_output_trees_redshifts`

**file:** `galacticus.output.merger_tree.rotation_curve.F90`

**Description:** Contains a module which handles outputting of rotation curve data to the GALACTICUS output file.

**Code lines:** 312

**module:** `galacticus_output_trees_rotation_curve`

**Description:** Handles outputting of rotation curve data to the GALACTICUS output file.

**Code lines:** 292

**Contained by:** file `galacticus.output.merger_tree.rotation_curve.F90`

**Modules used:** `iso_varying_string`

**Used by:** subroutine `count_properties` subroutine `establish_property_names` subroutine `galacticus_merger_tree_output`

**subroutine:** `galacticus_output_tree_rotation_curve`

**Description:** Store rotation curve properties in the GALACTICUS output file buffers.

**Code lines:** 58

**Contained by:** module `galacticus_output_trees_rotation_curve`

**Modules used:** `dark_matter_halo_scales` `galactic_structure_enclosed_masses` `galactic_structure_options` `galactic_structure_rotation_curves`
**19. Source Code Documentation**

```plaintext
galacticus_nodes                kind_numbers

**subroutine:** galacticus_output_tree_rotation_curve_initialize
**Description:** Initializes the module by determining whether or not rotation curve data should be output.
**Code lines:** 131
**Contained by:** module galacticus_output_trees_rotation_curve
**Modules used:**
galacticus_structure_options    galacticus_error
galacticus_nodes               input_parameters
stellar_luminosities_structure string_handling

**subroutine:** galacticus_output_tree_rotation_curve_names
**Description:** Set the names of rotation curve properties to be written to the Galacticus output file.
**Code lines:** 33
**Contained by:** module galacticus_output_trees_rotation_curve
**Modules used:**
galacticus_nodes               numerical_constants_astronomical

**subroutine:** galacticus_output_tree_rotation_curve_property_count
**Description:** Account for the number of rotation curve properties to be written to the Galacticus output file.
**Code lines:** 14
**Contained by:** module galacticus_output_trees_rotation_curve
**Modules used:**
galacticus_nodes

**type:** radiusspecifier
**Code lines:** 5
**Contained by:** module galacticus_output_trees_rotation_curve

**file:** galacticus.output.merger_tree.satellite_extrema.F90
**Description:** Contains a module which handles outputting of satellite orbital extremum data to the Galacticus output file.
**Code lines:** 247

**module:** galacticus_output_trees_satellite_extremum
**Description:** Handles outputting of satellite orbital extremum data to the Galacticus output file.
**Code lines:** 227
**Contained by:** file galacticus.output.merger_tree.satellite_extrema.F90
**Used by:**
subroutine count_properties      subroutine establish_property_names
subroutine galacticus_merger_tree_output

**subroutine:** galacticus_output_tree_satellite_extremum
**Description:** Store satellite orbital extremum properties in the Galacticus output file buffers.
**Code lines:** 65
**Contained by:** module galacticus_output_trees_satellite_extremum
**Modules used:**
galacticus_nodes               kind_numbers
kepler_orbits                  satellite_orbits

**subroutine:** galacticus_output_tree_satellite_extremum_initialize
**Description:** Initializes the module by determining whether or not satellite extremum data should be output.
```

1138
19.1. Program units

**Code lines:** 42  
**Contained by:** module `galacticus_output_trees_satellite_extremum`  
**Modules used:** `input_parameters`

**subroutine:** `galacticus_output_tree_satellite_extremum_names`  
**Description:** Set the names of satellite orbital extremum properties to be written to the GALACTICUS output file.  
**Code lines:** 69  
**Contained by:** module `galacticus_output_trees_satellite_extremum`  
**Modules used:** `galacticus_nodes`, `numerical_constants_astronomical`

**subroutine:** `galacticus_output_tree_satellite_extremum_property_count`  
**Description:** Account for the number of satellite orbital extremum properties to be written to the GALACTICUS output file.  
**Code lines:** 14  
**Contained by:** module `galacticus_output_trees_satellite_extremum`  
**Modules used:** `galacticus_nodes`

**file:** `galacticus.output.merger_tree.satellite_host.F90`  
**Description:** Contains a module which handles outputting of satellite host properties to the GALACTICUS output file.  
**Code lines:** 167  
**module:** `galacticus_output_trees_satellite_host`  
**Description:** Handles outputting of satellite host data to the GALACTICUS output file.  
**Code lines:** 147  
**Contained by:** file `galacticus.output.merger_tree.satellite_host.F90`  
**Used by:** subroutine `count_properties`, subroutine `establish_property_names`, subroutine `galacticus_merger_tree_output`  
**Modules used:** `galacticus_nodes`, `kepler_orbits`, `kind_numbers`, `satellite_orbits`

**subroutine:** `galacticus_output_tree_satellite_host`  
**Description:** Store satellite host halo properties in the GALACTICUS output file buffers.  
**Code lines:** 37  
**Contained by:** module `galacticus_output_trees_satellite_host`  
**Modules used:** `galacticus_nodes`, `kepler_orbits`, `kind_numbers`, `satellite_orbits`

**subroutine:** `galacticus_output_tree_satellite_host_initialize`  
**Description:** Initializes the module by determining whether or not satellite host data should be output.  
**Code lines:** 29  
**Contained by:** module `galacticus_output_trees_satellite_host`  
**Modules used:** `input_parameters`

**subroutine:** `galacticus_output_tree_satellite_host_names`  
**Description:** Set the names of satellite orbital extremum properties to be written to the GALACTICUS output file.  
**Code lines:** 30  
**Contained by:** module `galacticus_output_trees_satellite_host`  
**Modules used:** `galacticus_nodes`, `numerical_constants_astronomical`
subroutine: galacticus_output_tree_satellite_host_property_count
Description: Account for the number of satellite host properties to be written to the GALACTICUS output file.
Code lines: 14
Contained by: module galacticus_output_trees_satellite_host
Modules used: galacticus_nodes

file: galacticus.output.merger_tree.satellite_status.F90
Description: Contains a module which handles outputting of satellite status (i.e. whether orphaned or not) to the GALACTICUS output file.
Code lines: 181

module: galacticus_output_trees_satellite_status
Description: Handles outputting of satellite status (i.e. whether orphaned or not) to the GALACTICUS output file.
Code lines: 161
Contained by: file galacticus.output.merger_tree.satellite_status.F90
Used by: subroutine count_properties subroutine establish_property_names subroutine galacticus_merger_tree-output

subroutine: galacticus_output_tree_satellite_status
Description: Store satellite host halo properties in the GALACTICUS output file buffers.
Code lines: 45
Contained by: module galacticus_output_trees_satellite_status
Modules used: galacticus_nodes histories kind_numbers

subroutine: galacticus_output_tree_satellite_status_initialize
Description: Initializes the module by determining whether or not satellite host data should be output.
Code lines: 35
Contained by: module galacticus_output_trees_satellite_status
Modules used: galacticus_error galacticus_nodes input_parameters iso_varying_string

subroutine: galacticus_output_tree_satellite_status_names
Description: Set the names of satellite orbital extremum properties to be written to the GALACTICUS output file.
Code lines: 30
Contained by: module galacticus_output_trees_satellite_status
Modules used: galacticus_nodes numerical_constants_astronomical

subroutine: galacticus_output_tree_satellite_status_property_count
Description: Account for the number of satellite host properties to be written to the GALACTICUS output file.
Code lines: 14
Contained by: module galacticus_output_trees_satellite_status
Modules used: galacticus_nodes
**file: galacticus.output.merger_tree.star_formation.F90**

*Description:* Contains a module which handles computation and output of star formation histories for galaxies.

*Code lines:* 210

**module: galacticus_output_star_formation_histories**

*Description:* Handles computation and output of star formation histories for galaxies.

*Code lines:* 190

*Contained by:* file galacticus.output.merger_tree.star_formation.F90

*Modules used:* abundances_structure galacticus_nodes

history iso_varying_string kind_numbers

*Used by:* subroutine node_component_disk_- exponential_create subroutine node_component_disk_- exponential_rate_compute subroutine node_component_disk_- exponential_scale_set subroutine node_component_spheroid_- standard_initializer subroutine node_component_spheroid_- standard_rate_compute subroutine node_component_spheroid_- standard_scale_set subroutine node_component_spheroid_- standard_starformationhistory_- output

**subroutine: galacticus_output_star_formation_histories_initialize**

*Description:* Initialize the star formation histories module.

*Code lines:* 55

*Contained by:* module galacticus_output_star_formation_histories

*Modules used:* galacticus_error input_parameters starformationhistories_- starformationhistories_null metallicity_split

**subroutine: starformationhistory_create**

*Description:* Create any history required for storing the star formation history.

*Code lines:* 13

*Contained by:* module galacticus_output_star_formation_histories

**subroutine: starformationhistory_output**

*Description:* Output the star formation history for thisNode.

*Code lines:* 17

*Contained by:* module galacticus_output_star_formation_histories

**subroutine: starformationhistory_record**

*Description:* Record the star formation history for thisNode.

*Code lines:* 15

*Contained by:* module galacticus_output_star_formation_histories

**subroutine: starformationhistory_scales**
19. Source Code Documentation

**Description:** Set the scaling factors for error control on the absolute value of stellar population properties.

**Code lines:** 13

**Contained by:** module `galacticus_output_star_formation_histories`

**file:** `galacticus.output.merger_tree.star_formation.metallicity_split.F90`

**Description:** Contains a module which handles computation and output of star formation histories split by metallicity for galaxies.

**Code lines:** 478

**module:** `star_formation_histories_metallicity_split`

**Description:** Handles computation and output of star formation histories split by metallicity for galaxies.

**Code lines:** 458

**Contained by:** file `galacticus.output.merger_tree.star_formation.metallicity_split.F90`

**Used by:** subroutine `galacticus_output_star_formation_histories_initialize`

**subroutine:** `star_formation_histories_metallicity_split_initialize`

**Description:** Initializes the metallicity split star formation history module.

**Code lines:** 110

**Contained by:** module `star_formation_histories_metallicity_split`

**Modules used:** galacticus_error, input_parameters, iso_varying_string, memory_management, numerical_ranges

**subroutine:** `star_formation_history_create_metallicity_split`

**Description:** Create the history required for storing star formation history.

**Code lines:** 17

**Contained by:** module `star_formation_histories_metallicity_split`

**Modules used:** galacticus_error, galacticus_output_times, histories

**subroutine:** `star_formation_history_metallicity_split_make_history`

**Description:** Create the history required for storing star formation history.

**Code lines:** 138

**Contained by:** module `star_formation_histories_metallicity_split`

**Modules used:** galacticus_error, galacticus_output_times, histories, numerical_ranges

**subroutine:** `star_formation_history_output_metallicity_split`

**Description:** Output the star formation history for thisNode.

**Code lines:** 85

**Contained by:** module `star_formation_histories_metallicity_split`

**Modules used:** galacticus_hdf5, galacticus_nodes, galacticus_output_times, histories, iso_c_binding, iso_varying_string, kind_numbers, string_handling

**subroutine:** `star_formation_history_record_metallicity_split`

**Description:** Record the star formation history for thisNode.

1142
19.1. Program units

**Code lines:** 39  
**Contained by:** module `star_formation_histories_metallicity_split`  
**Modules used:** `abundances_structure` `arrays_search` `galacticus_nodes` `histories` `iso_c_binding`

**subroutine:** `star_formation_history_scales_metallicity_split`  
**Description:** Set the scalings for error control on the absolute values of star formation histories.  
**Code lines:** 28  
**Contained by:** module `star_formation_histories_metallicity_split`  
**Modules used:** `abundances_structure` `histories` `memory_management`

**type:** `timesteprange`  
**Code lines:** 4  
**Contained by:** module `star_formation_histories_metallicity_split`

**file:** `galacticus.output.merger_tree.star_formation.null.F90`  
**Description:** Contains a module which implements a null method for star formation histories.  
**Code lines:** 110

**module:** `star_formation_histories_null`  
**Description:** Implements a null method for star formation histories.  
**Code lines:** 90  
**Contained by:** file `galacticus.output.merger_tree.star_formation.null.F90`  
**Used by:** subroutine `galacticus_output_star_formation_histories_initialize`

**subroutine:** `star_formation_histories_null_initialize`  
**Description:** Initializes the metallicity split star formation history module.  
**Code lines:** 18  
**Contained by:** module `star_formation_histories_null`  
**Modules used:** `iso_varying_string`

**subroutine:** `star_formation_history_create_null`  
**Description:** Create the history required for storing star formation history.  
**Code lines:** 10  
**Contained by:** module `star_formation_histories_null`  
**Modules used:** `galacticus_nodes` `histories`

**subroutine:** `star_formation_history_output_null`  
**Description:** Output the star formation history for thisNode.  
**Code lines:** 16  
**Contained by:** module `star_formation_histories_null`  
**Modules used:** `galacticus_nodes` `histories` `iso_c_binding` `kind_numbers`

**subroutine:** `star_formation_history_record_null`  
**Description:** Record the star formation history for thisNode.
19. Source Code Documentation

**Code lines:** 14
**Contained by:** module `star_formation_histories_null`
**Modules used:** `abundances_structure` `galacticus_nodes` `histories`

**subroutine:** `star_formation_history_scales_null`
**Description:** Set the scalings for error control on the absolute values of star formation histories.
**Code lines:** 11
**Contained by:** module `star_formation_histories_null`
**Modules used:** `abundances_structure` `histories`

**file:** `galacticus.output.merger_tree.tree_indices.F90`
**Description:** Contains a module which handles outputting of tree index data to the GALACTICUS output file.
**Code lines:** 142

**module:** `galacticus_output_trees_tree_indices`
**Description:** Handles outputting of tree index data to the GALACTICUS output file.
**Code lines:** 122
**Contained by:** file `galacticus.output.merger_tree.tree_indices.F90`
**Modules used:** `galacticus_nodes`
**Used by:** subroutine `count_properties` subroutine `establish_property_names` subroutine `galacticus_merger_tree_output`
**subroutine:** `galacticus_output_tree_tree_indices_initialize`
**Description:** Initializes the module by determining whether or not tree index data should be output.
**Code lines:** 27
**Contained by:** module `galacticus_output_trees_tree_indices`
**Modules used:** `galacticus_error` `input_parameters`

**subroutine:** `galacticus_output_tree_tree_indices`
**Description:** Store link properties in the GALACTICUS output file buffers.
**Code lines:** 18
**Contained by:** module `galacticus_output_trees_tree_indices`
**Modules used:** `kind_numbers`

**subroutine:** `galacticus_output_tree_tree_indices_names`
**Description:** Set the names of tree index properties to be written to the GALACTICUS output file.
**Code lines:** 27
**Contained by:** module `galacticus_output_trees_tree_indices`

**subroutine:** `galacticus_output_tree_tree_indices_property_count`
**Description:** Account for the number of link properties to be written to the GALACTICUS output file.
**Code lines:** 12
**Contained by:** module `galacticus_output_trees_tree_indices`

**file:** `galacticus.output.merger_tree.velocity Dispersion.F90`
**Description:** Contains a module which handles outputting of velocity dispersion data to the GALACTICUS output file.
module: galacticus_output_trees_velocity_dispersion
Description: Handles outputting of velocity dispersion data to the GALACTICUS output file.
Code lines: 654
Contained by: file galacticus.output.merger_tree.velocity_dispersion.F90
Modules used: galacticus_nodes iso_varying_string
Used by: subroutine count_properties subroutine establish_property_names subroutine galacticus_merger_tree_output

subroutine: galacticus_output_tree_velocity_dispersion
Description: Store velocity dispersion properties in the GALACTICUS output file buffers.
Code lines: 145
Contained by: module galacticus_output_trees_velocity_dispersion
Modules used: dark_matter_halo_scales fgsl
galactic_structure_enclosed_masses galactic_structure_options
galactic_structure_velocity dispersions
kind_numbers iso_c_binding
numerical_integration

subroutine: galacticus_output_tree_velocity_dispersion_initialize
Description: Initializes the module by determining whether or not velocity dispersion should be output.
Code lines: 161
Contained by: module galacticus_output_trees_velocity_dispersion
Modules used: galactic_structure_options galacticus_error
input_parameters stellar_luminosities_structure
string_handling

subroutine: galacticus_output_tree_velocity_dispersion_names
Description: Set the names of velocity dispersion properties to be written to the GALACTICUS output file.
Code lines: 32
Contained by: module galacticus_output_trees_velocity_dispersion
Modules used: numerical_constants_astronomical

subroutine: galacticus_output_tree_velocity_dispersion_property_count
Description: Account for the number of velocity dispersion properties to be written to the GALACTICUS output file.
Code lines: 13
Contained by: module galacticus_output_trees_velocity_dispersion

function: galacticus_output_trees_line_of_sight_velocity_dispersion
Description: Compute the line-of-sight velocity dispersion at the given radius.
Code lines: 22
Contained by: module galacticus_output_trees_velocity_dispersion
Modules used: fgsl iso_c_binding numerical_integration
function: galacticus_output_trees_velocity_dispersion_density_integrand
Description: Integrand function used for computing line-of-sight velocity dispersions.
Code lines: 15
Contained by: module galacticus_output_trees_velocity_dispersion
Modules used: galactic_structure_densities iso_c_binding

function: galacticus_output_trees_vlcty_dsprsn_vlcty_dnsty_intgrnd
Description: Integrand function used for computing line-of-sight velocity dispersions. Specifically, we wish to evaluate the integral:
\[
\int_{r_i}^{r_o} \sigma^2(r) \rho(r) \frac{r}{\sqrt{r^2 - r_i^2}} dr,
\]
where \( r_i \) is the impact parameter, \( r_o \) is an outer radius at which we assume \( \rho(r_o)\sigma^2(r_o) = 0 \) (i.e. it is the radius at which we begin integrating the Jeans equation), \( \rho(r) \) is density, and \( \sigma(r) \) is the velocity dispersion at radius \( r \). Assuming spherical symmetry and isotropic velocity dispersion, the Jeans equation tells us
\[
\rho(r)\sigma^2(r) = \int_r^{r_o} \frac{GM(<r')}{r'^2} \rho(r') dr',
\]
where \( G \) is the gravitational constant, and \( M(<r) \) is the total mass contained within radius \( r \). Equation (19.12) can then be simplified using integration by parts to give:
\[
\left[ \sigma^2(r) \rho(r) \sqrt{r^2 - r_i^2} \right]_{r_i}^{r_o} + \int_{r_i}^{r_o} \frac{d}{dr} \left[ \sigma^2(r) \rho(r) \right] \sqrt{r^2 - r_i^2} dr.
\]
The first term is zero at both limits (due to the constraint \( \rho(r_o)\sigma^2(r_o) = 0 \) at \( r_o \) and due to \( \sqrt{r^2 - r_i^2} = 0 \) at \( r_i \)), and the second term can be simplified using eqn. (19.13) to give
\[
\int_{r_i}^{r_o} \frac{GM(<r)}{r^2} \rho(r) \sqrt{r^2 - r_i^2} dr.
\]

function: galacticus_output_trees_vlcty_dsprsn_lambdar_intgrnd1
Description: Integrand function used for integrating line-of-sight surface density dispersion over area.
Code lines: 15
Contained by: module galacticus_output_trees_velocity_dispersion
Modules used: galactic_structure_densities iso_c_binding

function: galacticus_output_trees_vlcty_dsprsn_dnsty_srfc_intgrnd
Description: Integrand function used for integrating the $\lambda_R$ statistic of Cappellari et al. [2007]. In this case we want to evaluate

$$\int_0^r 2\pi r'\Sigma(r')\sqrt{\sigma^2(r') + V^2(r')}dr',$$  \hspace{1cm} (19.16)

where $\Sigma(r)$ is the projected surface density (in mass or light) of the galaxy at radius $r$, $\sigma^2(r)$ is the measured velocity dispersion and $V(r)$ the measured rotation speed. Assuming that the selected component is purely dispersion dominated with velocity dispersion $\sigma_s(r)$, and that rotation is present in only the disk component with rotation curve $V_d(r)$ then we can model the velocity distribution, $P(V)$, at $r$ as the sum of a Gaussian of width $\sigma_s(r)$ and normalized area $\Sigma_s(r)$, and a delta function at $V_d(r)$ with normalized area $\Sigma_d(r)$. The measured rotation speed is then:

$$V(r) = \frac{\int_{-\infty}^{+\infty} P(V)\mathcal{V}d\mathcal{V}}{\int_{-\infty}^{+\infty} P(V)d\mathcal{V}} = \frac{\Sigma_d(r)V_d(r)}{[\Sigma_d(r) + \Sigma_s(r)]},$$  \hspace{1cm} (19.17)

and the measured velocity dispersion is:

$$\sigma^2(r) = \frac{\int_{-\infty}^{+\infty} P(V)[V - V(r)]^2d\mathcal{V}}{\int_{-\infty}^{+\infty} P(V)d\mathcal{V}} = \frac{\Sigma_s(r)[\sigma^2_s(r)] + \Sigma_d(r)[V_d(r) - V(r)]^2}{[\Sigma_d(r) + \Sigma_s(r)]^2}.$$  \hspace{1cm} (19.18)

Code lines: 59

Contained by: module galacticus_output_trees_velocity_dispersion
function: galacticus_output_trees_vlcty_dsprsn_lambdar_intgrnd2
Description: Integrand function used for integrating the $\lambda_R$ statistic of Cappellari et al. [2007]. In this case we want to evaluate
\[
\int_0^r 2\pi r' \Sigma(r') V(r') dr',
\]
where $\Sigma(r)$ is the projected surface density (in mass or light) of the galaxy at radius $r$, and $V(r)$ the measured rotation speed. Assuming that the selected component is purely dispersion dominated with velocity dispersion $\sigma_s(r)$, and that rotation is present in only the disk component with rotation curve $V_d(r)$ then we can model the velocity distribution, $P(V)$, at $r$ as the sum of a Gaussian of width $\sigma_s(r)$ and normalized area $\Sigma_s(r)$, and a delta function at $V_d(r)$ with normalized area $\Sigma_d(r)$. The measured rotation speed is then:
\[
V(r) = \frac{\int_{-\infty}^{+\infty} P(V) V dV}{\int_{-\infty}^{+\infty} P(V) dV} = \frac{\Sigma_d(r) V_d(r)}{[\Sigma_d(r) + \Sigma_s(r)]},
\]
(19.20)

Code lines: 36
Contained by: module galacticus_output_trees_velocity_dispersion
Modules used: fgs1
    galactic_structure_rotation_curves
    galactic_structure_surface_densities
    iso_c_binding
    numerical_integration

function: galacticus_output_trees_vlcty_dsprsn_vlcty_dnsty_srfc_intgrnd
Description: Integrand function used for integrating line-of-sight velocity dispersion over surface density.

Code lines: 16
Contained by: module galacticus_output_trees_velocity_dispersion
Modules used: galactic_structure_densities
galactic_structure_velocity_-dispersions
iso_c_binding
numerical_integration

type: radiusspecifier
Code lines: 5
Contained by: module galacticus_output_trees_velocity_dispersion

function: spherical_shell_solid_angle_in_cylcinder
Description: Computes the solid angle of a spherical shell of given radius that lies within a cylinder of radius radiusImpact.

Code lines: 16
Contained by: module galacticus_output_trees_velocity_dispersion
Modules used: numerical_constants_math

file: galacticus.output.merger_tree.virial.F90
19.1. Program units

**Description:** Contains a module which handles outputting of node virial data to the GALACTICUS output file.

**Code lines:** 165

**module:** galacticus_output_trees_virial

**Description:** Handles outputting of node virial data to the GALACTICUS output file.

**Code lines:** 145

**Contained by:** file galacticus.output.merger_tree.virial.F90

**Used by:** subroutine count_properties subroutine establish_property_names subroutine galacticus_merger_tree_output

**subroutine:** galacticus_output_tree_virial

**Description:** Store virial properties in the GALACTICUS output file buffers.

**Code lines:** 25

**Contained by:** module galacticus_output_trees_virial

**Modules used:** dark_matter_halo_scales galacticus_nodes kind_numbers

**subroutine:** galacticus_output_tree_virial_initialize

**Description:** Initializes the module by determining whether or not virial data should be output.

**Code lines:** 27

**Contained by:** module galacticus_output_trees_virial

**Modules used:** input_parameters

**subroutine:** galacticus_output_tree_virial_names

**Description:** Set the names of virial properties to be written to the GALACTICUS output file.

**Code lines:** 42

**Contained by:** module galacticus_output_trees_virial

**Modules used:** galacticus_nodes numerical_constants_astronomical

**subroutine:** galacticus_output_tree_virial_property_count

**Description:** Account for the number of virial properties to be written to the GALACTICUS output file.

**Code lines:** 14

**Contained by:** module galacticus_output_trees_virial

**Modules used:** galacticus_nodes

**file:** galacticus.output.times.F90

**Description:** Contains a module which provides output times.

**Code lines:** 210

**module:** galacticus_output_times

**Description:** Provides output times.

**Code lines:** 190

**Contained by:** file galacticus.output.times.F90

**Modules used:** iso_c_binding

**Used by:** subroutine galacticus_output_analysis_mass_dpndnt(sz_dstrbtins) subroutine galacticus_output_analysis_mass_functions
subroutine galacticus_output_tree_descendents subroutine galacticus_output_tree_final_descendents subroutine star_formation_history_create_metallicity_split subroutine star_formation_history_output_metallicity_split subroutine intergalactic_medium_state_internal_initialize subroutine merger_tree_read_initialize function galacticus_task_evolve_tree subroutine node_component_age_statistics_standard_rate_compute subroutine node_component_merging_statistics_recent_node_merger subroutine node_component_merging_statistics_recent_output subroutine node_component_merging_statistics_recent_output special_cases subroutine stellar_luminosities_special_cases function radiation_intergalactic_background_internal_update

function: galacticus_next_output_time

Description: Returns the time of the next output after currentTime.
Code lines: 25
Contained by: module galacticus_output_times
Modules used: arrays_search kind_numbers

function: galacticus_output_redshift

Description: Returns the redshift of the output indexed by iOutput.
Code lines: 15
Contained by: module galacticus_output_times

function: galacticus_output_time

Description: Returns the time of the output indexed by iOutput.
Code lines: 15
Contained by: module galacticus_output_times

function: galacticus_output_time_count

Description: Return the number of outputs.
Code lines: 11
Contained by: module galacticus_output_times

function: galacticus_output_time_index

Description: Returns the index of the output given the corresponding time.
Code lines: 16
Contained by: module galacticus_output_times
Modules used: arrays_search galacticus_error kind_numbers numerical_comparison

1150
19.1. Program units

function: galacticus_previous_output_time
Description: Returns the time of the previous output prior to currentTime.
Code lines: 19
Contained by: module galacticus_output_times
Modules used: arrays_search

subroutine: output_times_initialize
Description: Initialize the output times.
Code lines: 59
Contained by: module galacticus_output_times
Modules used: cosmology_functions histories input_parameters memory_management sort

file: galacticus.output.version.F90
Description: Contains a module which implements writing of the version number and run time to the
Galacticus output file.
Code lines: 110

module: galacticus_versioning
Description: Implements writing of the version number and run time to the GALACTICUS output file.
Code lines: 90
Contained by: file galacticus.output.version.F90
Used by: subroutine galacticus_output_open_file module input_parameters

function: galacticus_version
Description: Returns a string describing the version of GALACTICUS.
Code lines: 10
Contained by: module galacticus_versioning
Modules used: iso_varying_string string_handling

subroutine: galacticus_version_output
Description: Output version information to the main output file.
Code lines: 55
Contained by: module galacticus_versioning
Modules used: dates_and_times file_utilities fox_dom fox_utils galacticus_error galacticus_hdf5 io_hdf5 io_xml iso_varying_string

file: galacticus.state.F90
Description: Contains a module which implements storage and recovery of the Galacticus internal state.
Used for restoring random number generator sequences for example.
Code lines: 367

module: galacticus_state
Description: Implements storage and recovery of the Galacticus internal state. Used for restoring random
number generator sequences for example.
19. Source Code Documentation

**Code lines:** 346
**Contained by:** file `galacticus.state.F90`
**Modules used:** `iso_varying_string` tables
**Used by:** subroutine `merger_tree_build_do`
subroutine `merger_tree_state_restore`
**subroutine: galacticus_state_retrieve**
**Description:** Retrieve the interal state.
**Code lines:** 119
**Contained by:** module `galacticus_state`
**Modules used:**
- `black_hole_binary_recoil_velocities_-standard` `cosmological_mass_variance_filtered_-power_spectrum`
- `cosmology_functions` `critical_overdensity`
- `dark_matter_halo_scales` `dark_matter_profiles`
- `fgsl` `halo_spin_distributions_bett2007`
- `halo_spin_distributions_lognormal` `histories`
- `hot_halo_mass_distributions_core_-radii` `linear_growth`
- `linear_growth_simple` `mass_distributions`
- `merger_tree_build_cole2000` `merger_tree_read_state`
- `node_component_disk_exponential_data` `primordial_power_spectra`
- `primordial_power_spectrum_power-law` `satellite_merging_timescales`
- `sphericalCollapse_matter_dark_energy` `sphericalCollapse_matter_lambda`
- `string_handling` `transfer_function_bbks`
- `transfer_function_eisenstein_hu` `transfer_functions`
- `virial_density_contrast` `virial_orbits_benson2005`
- `virial_orbits_wetzel2010`

**subroutine: galacticus_state_snapshot**
**Description:** Take a snapshot of the internal state.
**Code lines:** 42
**Contained by:** module `galacticus_state`
**Modules used:**
- `black_hole_binary_recoil_velocities_-standard`
- `halo_spin_distributions_lognormal`
- `satellite_merging_timescales`
- `virial_orbits_wetzel2010`
- `halo_spin_distributions_bett2007`
- `merger_tree_build_cole2000`
- `virial_orbits_benson2005`

**subroutine: galacticus_state_store**
**Description:** Store the internal state.
**Code lines:** 116
**Contained by:** module `galacticus_state`
**Modules used:**
- `black_hole_binary_recoil_velocities_-standard` `cosmological_mass_variance_filtered_-power_spectrum`
- `cosmology_functions` `dark_matter_halo_scales`
- `dark_matter_profiles` `fgsl`
- `halo_spin_distributions_bett2007` `halo_spin_distributions_lognormal`
- `histories` `hot_halo_mass_distributions_core_-radii`
subroutine: state_initialize
Description: Initialize the state module by getting the name of the file to which states should be stored and whether or not we are to retrieve a state.
Code lines: 41
Contained by: module galacticus_state
Modules used: input_parameters

file: galacticus.tasks.F90
Description: Contains a module which defines and keeps track of the current task in GALACTICUS.
Code lines: 73

module: galacticus_tasks
Description: Defines and keeps track of the current task in GALACTICUS.
Code lines: 53
Contained by: file galacticus.tasks.F90
Used by: program galacticus

subroutine: galacticus_task_do
Description: Performs GALACTICUS tasks.
Code lines: 40
Contained by: module galacticus_tasks
Modules used: galacticus_tasks_basic galacticus_tasks_evolve_tree

file: galacticus.tasks.basic.F90
Code lines: 90

module: galacticus_tasks_basic
Code lines: 72
Contained by: file galacticus.tasks.basic.F90
Modules used: galacticus_display
Used by: subroutine galacticus_task_do

function: galacticus_task_end
Code lines: 19
Contained by: module galacticus_tasks_basic
Modules used: galacticus_output_merge_tree galacticus_output_open

function: galacticus_task_start
Code lines: 34
Contained by: module galacticus_tasks_basic
19. Source Code Documentation

**Modals used:**
galacticus_output_open
input_parameters

**file:** galacticus.tasks.evolve_tree.F90

*Description:* Contains a module which implements the task of evolving merger trees.

*Code lines:* 556

**module:** galacticus_tasks_evolve_tree

*Description:* Implements the task of evolving merger trees.

*Code lines:* 536

*Contained by:* file galacticus.tasks.evolve_tree.F90

*Used by:* subroutine galacticus_task_do

**function:** galacticus_task_evolve_tree

*Description:* Evolves the complete set of merger trees as specified.

*Code lines:* 462

*Contained by:* module galacticus_tasks_evolve_tree

*Modules used:*
galacticus_display
galacticus_error
galacticus_merger_tree_output_
filter_lightcones
galacticus_nodes
galacticus_output_merge_tree
galacticus_output_times
intergalactic_medium_state_internal_
evolver
iso_varying_string
merger_tree_dump_structure
merger_tree_output_structure
merger_trees_evolve
merger_trees_prune_branches
merger_trees_prune_non_essential
merger_trees_write
radiation_intergalactic_background_
internal
string_handling

galacticus_meta_tree_timing
iso_c_binding

memory_management
merger_tree_mass_accretion_history
merger_trees_conditional_mass_
function
merger_trees_monotonic_mass_growth
merger_trees_prune_hierarchy
merger_trees_regrid_times
node_components
semaphores

system_load

**subroutine:** get_tree

*Description:* Get a tree to process.

*Code lines:* 46

*Contained by:* module galacticus_tasks_evolve_tree

*Modules used:*
galacticus_meta_tree_timing

node_component_hot_halo_cold_mode
node_components
node_component_satellite_orbiting

**file:** geometry.coordinate_systems.F90

*Description:* Contains a module which implements calculations related to coordinate systems and transformations.

*Code lines:* 103

**module:** coordinate_systems

---

1154
19.1. Program units

Description: Implements calculations related to coordinate systems and transformations.
Code lines: 83
Contained by: file `geometry.coordinate_systems.F90`
Used by: function `galactic_structure_density`
function `galactic_structure_surface_density`
function `node_component_disk_exponential_density`
program `test_coordinate_systems`

function: coordinates_cartesian_to_cylindrical
Description: Convert \((x, y, z)\) in Cartesian coordinates into \((r, \phi, z)\) in cylindrical coordinates, with \(\phi = 0\) corresponding to the \(x\)-axis.
Code lines: 13
Contained by: module `coordinate_systems`

function: coordinates_cartesian_to_spherical
Description: Convert \((x, y, z)\) in Cartesian coordinates into \((r, \theta, \phi)\) in spherical coordinates, with \(\theta = 0\) corresponding to the \(z\)-axis and \(\phi = 0\) corresponding to the \(x\)-axis.
Code lines: 20
Contained by: module `coordinate_systems`

function: coordinates_cylindrical_to_spherical
Description: Convert \((R, \phi, z)\) in cylindrical coordinates into \((r, \theta, \phi)\) in spherical coordinates, with \(\phi = 0\) corresponding to the \(x\)-axis.
Code lines: 20
Contained by: module `coordinate_systems`

function: coordinates_spherical_to_cylindrical
Description: Convert \((r, \theta, \phi)\) in spherical coordinates into \((R, \phi, z)\) in cylindrical coordinates, with \(\phi = 0\) corresponding to the \(x\)-axis.
Code lines: 14
Contained by: module `coordinate_systems`

file: `geometry.surveys.Caputi-2011-UKIDSS-UDS.F90`
Description: Contains a module which implements the survey geometry used by Caputi et al. [2011].
Code lines: 200

module: `geometry_surveys_caputi_2011_ukidss_uds`
Description: Implements the survey geometry used by Caputi et al. [2011].
Code lines: 180
Contained by: file `geometry.surveys.Caputi-2011-UKIDSS-UDS.F90`
Used by: subroutine `geometry_surveys_initialize`

function: `geometry_survey_distance_maximum_caputi_2011_ukidss_uds`
Description: Compute the maximum distance at which a galaxy is visible.
Code lines: 18
Contained by: module `geometry_surveys_caputi_2011_ukidss_uds`
Modules used: `cosmology_functions`
`cosmology_functions_options`

function: `geometry_survey_solid_angle_caputi_2011_ukidss_uds`
19. Source Code Documentation

**Description:** Return the solid angle of the Caputi et al. [2011] sample. Computed from survey mask (see constraints/dataAnalysis/stellarMassFunctions_UKIDSS_UDS_z3-5/surveyGeometryRandoms.pl).

**Code lines:** 8
**Contained by:** module geometry_surveys_caputi_2011_ukidss_uds

**function:** geometry_survey_volume_maximum_caputi_2011_ukidss_uds

**Description:** Compute the maximum volume in which a galaxy of given mass could have been observed.

**Code lines:** 8
**Contained by:** module geometry_surveys_caputi_2011_ukidss_uds

**subroutine:** geometry_survey_window_functions_caputi_2011_ukidss_uds

**Description:** Compute the window function for the survey.

**Code lines:** 107
**Contained by:** module geometry_surveys_caputi_2011_ukidss_uds

**Modules used:**
- cosmology_functions
- fftw3
- fgsl
- galacticus_error
- galacticus_input_paths
- iso_c_binding
- io_hdf5
- memory_management
- numerical_constants_math
- pseudo_random
- string_handling
- system_command
- vectors

**subroutine:** geometry_surveys_caputi_2011_ukidss_uds_initialize

**Description:** Initializes the “Caputi-2011-UKIDSS-UDS” survey geometry module.

**Code lines:** 18
**Contained by:** module geometry_surveys_caputi_2011_ukidss_uds

**Modules used:**
- input_parameters
- iso_varying_string

**file:** geometry.surveys.F90

**Description:** Contains a module which implements geometries of galaxy surveys.

**Code lines:** 160

**module:** geometry_surveys

**Description:** Implements geometries of galaxy surveys.

**Code lines:** 140
**Contained by:** file geometry.surveys.F90

**Modules used:**
- iso_varying_string

**Used by:**
- program conditional_mass_function
- subroutine mass_function_covariance_matrix

**function:** geometry_survey_distance_maximum

**Description:** Returns the maximum distance (in Mpc) at which a galaxy of the specified mass (in $M_\odot$) could be detected.

**Code lines:** 12
**Contained by:** module geometry_surveys
19.1. Program units

function: geometry_survey_solid_angle
Description: Returns the solid angle (in steradians) of the survey.
Code lines: 11
Contained by: module geometry_surveys

function: geometry_survey_volume_maximum
Description: Returns the maximum volume (in Mpc$^3$) at which a galaxy of the specified mass (in $M_\odot$) could be detected.
Code lines: 12
Contained by: module geometry_surveys

subroutine: geometry_survey_window_functions
Description: Returns the window functions on a grid of the specified size (gridCount cells in each dimension) for galaxies of the specified mass1 and mass2 (in $M_\odot$). The boxLength should be set to an appropriate value to fully enclose (with sufficient buffering to allow for Fourier transformation) the two window functions.
Code lines: 19
Contained by: module geometry_surveys
Modules used: iso_c_binding

subroutine: geometry_surveys_initialize
Description: Initialize the spheroid star formation timecale module.
Code lines: 55
Contained by: module geometry_surveys
Modules used: galacticus_error geometry_surveys_caputi_2011_ukidss_uds geometry_surveys_li_white_2009_sdss geometry_surveys_martin_2010_alfalfa input_parameters

file: geometry_surveys.Li-White-2009-SDSS.F90
Description: Contains a module which implements the survey geometry used by Li and White [2009].
Code lines: 210

module: geometry_surveys_li_white_2009_sdss
Description: Implements the survey geometry used by Li and White [2009].
Code lines: 190
Contained by: file geometry_surveys.Li-White-2009-SDSS.F90
Used by: subroutine geometry_surveys_initialize

function: geometry_survey_distance_maximum_li_white_2009_sdss
Description: Compute the maximum distance at which a galaxy is visible.
Code lines: 18
Contained by: module geometry_surveys_li_white_2009_sdss
Modules used: cosmology_functions cosmology_functions_options

function: geometry_survey_solid_angle_li_white_2009_sdss
Description: Return the solid angle of the Li and White [2009] sample.
Code lines: 7
19. Source Code Documentation

Contained by: module geometry_surveys_li_white_2009_sdss

function: geometry_survey_volume_maximum_li_white_2009_sdss
Description: Compute the maximum volume in which a galaxy of given mass could have been observed.
Code lines: 8
Contained by: module geometry_surveys_li_white_2009_sdss

subroutine: geometry_survey_window_functions_li_white_2009_sdss
Description: Compute the window function for the survey.
Code lines: 118
Contained by: module geometry_surveys_li_white_2009_sdss
Modules used:
cosmology_functions
fgsl
galacticus_display
galacticus_input_paths
iso_varying_string
meshes
pseudo_random
system_command

subroutine: geometry_surveys_li_white_2009_sdss_initialize
Description: Initializes the “Li-White-2009-SDSS” survey geometry module.
Code lines: 18
Contained by: module geometry_surveys_li_white_2009_sdss
Modules used: input_parameters iso_varying_string

file: geometry.surveys.Martin-2010-ALFALFA.F90
Description: Contains a module which implements the survey geometry used by Martin et al. [2010].
Code lines: 231

module: geometry_surveys_martin_2010_alfalfa
Description: Implements the survey geometry used by Martin et al. [2010].
Code lines: 211
Contained by: file geometry.surveys.Martin-2010-ALFALFA.F90
Used by: subroutine geometry_surveys_initialize

function: geometry_survey_distance_maximum_martin_2010_alfalfa
Description: Compute the maximum distance at which a galaxy is visible.
Code lines: 35
Contained by: module geometry_surveys_martin_2010_alfalfa
Modules used: cosmology_parameters

function: geometry_survey_solid_angle_martin_2010_alfalfa
Description: Return the solid angle of the Martin et al. [2010] sample.
Code lines: 7
Contained by: module geometry_surveys_martin_2010_alfalfa

function: geometry_survey_volume_maximum_martin_2010_alfalfa
Description: Compute the maximum volume in which a galaxy of given mass could have been observed.
19.1. Program units

Code lines: 8
Contained by: module geometry_surveys_martin_2010_alfalfa

subroutine: geometry_survey_window_functions_martin_2010_alfalfa
Description: Compute the window function for the survey.
Code lines: 119
Contained by: module geometry_surveys_martin_2010_alfalfa
Modules used:
- cosmology_functions
- fftw3
- file_utilities
- galacticus_error
- iso_c_binding
- memory_management
- numerical_constants_astronomical
- pseudo_random
- system_command

subroutine: geometry_surveys_martin_2010_alfalfa_initialize
Description: Initializes the “Martin-2010-ALFALFA” survey geometry module.
Code lines: 18
Contained by: module geometry_surveys_martin_2010_alfalfa
Modules used:
- input_parameters

file: halo_mass_functions.tasks.F90
Description: Contains a module which implements calculations of halo mass functions and related properties for output.
Code lines: 321

module: halo_mass_function_tasks
Description: Implements calculations of halo mass functions and related properties for output.
Code lines: 301
Contained by: file halo_mass_functions.tasks.F90
Modules used:
- io_hdf5

subroutine: halo_mass_function_close_file
Description: Close the output file for halo mass function data.
Code lines: 6
Contained by: module halo_mass_function_tasks

subroutine: halo_mass_function_compute
Description: Computes mass functions and related properties for output.
Code lines: 187
Contained by: module halo_mass_function_tasks
Modules used:
- cosmology_functions
- dark_matter_halo_biases
- dark_matter_profile_scales
- galacticus_calculations_resets

file: geometry_surveys_martin_2010_alfalfa
Description: Computes the window function for the survey.
Code lines: 119
Contained by: module geometry_surveys_martin_2010_alfalfa

Modules used:
- cosmology_functions
- fftw3
- file_utilities
- galacticus_error
- iso_c_binding
- memory_management
- numerical_constants_astronomical
- pseudo_random
- system_command

subroutine: geometry_surveys_martin_2010_alfalfa_initialize
Description: Initializes the “Martin-2010-ALFALFA” survey geometry module.
Code lines: 18
Contained by: module geometry_surveys_martin_2010_alfalfa
Modules used:
- input_parameters

file: halo_mass_functions.tasks.F90
Description: Contains a module which implements calculations of halo mass functions and related properties for output.
Code lines: 321

module: halo_mass_function_tasks
Description: Implements calculations of halo mass functions and related properties for output.
Code lines: 301
Contained by: file halo_mass_functions.tasks.F90
Modules used:
- io_hdf5

subroutine: halo_mass_function_close_file
Description: Close the output file for halo mass function data.
Code lines: 6
Contained by: module halo_mass_function_tasks

subroutine: halo_mass_function_compute
Description: Computes mass functions and related properties for output.
Code lines: 187
Contained by: module halo_mass_function_tasks
Modules used:
- cosmology_functions
- dark_matter_halo_biases
- dark_matter_profile_scales
- galacticus_calculations_resets

1159
19. Source Code Documentation

```
galacticus_nodes   halo_mass_function
input_parameters  iso_varying_string
linear_growth     memory_management
node_components   numerical_ranges
power_spectra     virial_density_contrast

subroutine: halo_mass_function_open_file
Description: Open the output file for halo mass function data.
Code lines: 14
Contained by: module halo_mass_function_tasks
Modules used: hdf5 iso_varying_string

subroutine: halo_mass_function_output
Description: Outputs halo mass function data.
Code lines: 65
Contained by: module halo_mass_function_tasks
Modules used: numerical_constants_astronomical

Description: Contains a module which implements the Behroozi et al. [2010] fitting function descriptions
of the conditional mass function.
Code lines: 335

module: conditional_mass_functions_behroozi2010
Description: Implements the Behroozi et al. [2010] fitting function descriptions of the conditional mass
function.
Code lines: 315
Contained by: file halo_model.conditional_mass_function.Behroozi2010.F90
Modules used: tables
Used by: subroutine conditional_mass_functions_-
initialize

subroutine: conditional_mass_functions_behroozi2010_initialize
Description: Initializes the “Behroozi2010” conditional mass function method.
Code lines: 161
Contained by: module conditional_mass_functions_behroozi2010
Modules used: input_parameters iso_varying_string

function: cumulative_conditional_mass_function_behroozi2010
Description: Computes the cumulative conditional mass function, \( N(M_\star|M_{\text{halo}}) \equiv \phi(M_\star|M_{\text{halo}}) \) using
the fitting formula of Behroozi et al. [2010].
Code lines: 13
Contained by: module conditional_mass_functions_behroozi2010

subroutine: cumulative_conditional_mass_function_compute
Description: Computes the cumulative conditional mass function, \( N(M_\star|M_{\text{halo}}) \equiv \phi(M_\star|M_{\text{halo}}) \) using
the fitting formula of Behroozi et al. [2010].
Code lines: 63
Contained by: module conditional_mass_functions_behroozi2010
```
function: cumulative_conditional_mass_function_var_behroozi2010
Description: Computes the variance in the cumulative conditional mass function, \( \langle N(M|M_{\text{halo}}) \rangle \equiv \phi(M|M_{\text{halo}}) \) using the fitting formula of Behroozi et al. [2010]. Assumes that the number of satellite galaxies is Poisson distributed, while the number of central galaxies follows a Bernoulli distribution, and that the numbers of satellites and centrals are uncorrelated.
Code lines: 18
Contained by: module conditional_mass_functions_behroozi2010

function: fshmrinverse
Description: The median mass vs. halo mass relation functional form from Behroozi et al. [2010].
Code lines: 17
Contained by: module conditional_mass_functions_behroozi2010

file: halo_model.conditional_mass_function.F90
Description: Contains a module which implements empirical models of conditional mass functions.
Code lines: 131

module: conditional_mass_functions
Description: Implements empirical models of conditional mass functions.
Code lines: 111
Contained by: file halo_model.conditional_mass_function.F90
Modules used: iso_varying_string
Used by: program conditional_mass_function function mass_function_halo_mass_integrand
function xi_integrand function stellar_mass_function_integrand
function halo_occupancy_integrand function mass_function_integrand_i

subroutine: conditional_mass_functions_initialize
Description: Initialize the conditional mass function module.
Code lines: 52
Contained by: module conditional_mass_functions
Modules used: conditional_mass_functions_behroozi2010 galacticus_error
input_parameters

function: cumulative_conditional_mass_function
Description: Returns the cumulative conditional mass function at a mass of mass in a halo of mass massHalo.
Code lines: 12
Contained by: module conditional_mass_functions

function: cumulative_conditional_mass_function_variance
Description: Returns the cumulative conditional mass function at a mass of mass in a halo of mass massHalo.
Code lines: 12
Contained by: module conditional_mass_functions

file: hot_halo.cold_mode.density_profile.cored_isothermal.core_radius.F90
19. Source Code Documentation

**Description:** Contains a module which implements calculations of the core radius in cored isothermal cold mode hot halo profiles.

**Code lines:** 105

**module:** hot_halo_cold_mode_density_cored_isothermal_core_radii

**Description:** Implements calculations of the core radius in cored isothermal cold mode hot halo profiles.

**Code lines:** 85

**Contained by:** file hot_halo.cold_mode.density_profile.cored_isothermal.core_radius.F90

**Modules used:** galacticus_nodes iso_varying_string

**Used by:**
- function node_component_hot_halo_cold_mode_density_task
- function node_component_hot_halo_cold_mode_enclosed_mass_task

**subroutine:** hot_halo_cold_mode_density_ciso_corer_init

**Description:** Initialize the cored isothermal cold mode hot halo profile core radius module.

**Code lines:** 52

**Contained by:** module hot_halo_cold_mode_density_cored_isothermal_core_radii

**Modules used:** galacticus_error hot_halo_cold_mode_density_ciso_corer_virial_fraction

**input_parameters**

**function:** hot_halo_cold_mode_density_ciso_corer_vfrac

**Description:** Returns the radius (in Mpc) of the core in a cored isothermal hot halo density profile. Assumes that the radius is a fixed fraction of the halo virial radius.

**Code lines:** 13

**Contained by:** module hot_halo_cold_mode_density_ciso_corer_virial_fraction

**file:** hot_halo.cold_mode.density_profile.cored_isothermal.core_radius.virial_radius.F90

**Description:** Contains a module which implements a calculation of the core radius in the cold mode hot halo density profile that is a fixed fraction of the virial radius.

**Code lines:** 76

**module:** hot_halo_cold_mode_density_ciso_corer_virial_fraction

**Description:** Implements a calculation of the core radius in the cold mode hot halo density profile that is a fixed fraction of the virial radius.

**Code lines:** 55

**Contained by:** file hot_halo.cold_mode.density_profile.cored_isothermal.core_radius.virial_radius.F90

**Used by:** subroutine hot_halo_cold_mode_density_ciso_corer_vf_initialize

**subroutine:** hot_halo_cold_mode_density_ciso_corer_vf_initialize

**Description:** Initializes the “virial radius fraction” cored isothermal hot halo profile core radius module.

**Code lines:** 23

**Contained by:** module hot_halo_cold_mode_density_ciso_corer_virial_fraction

**Modules used:** input_parameters iso_varying_string

**function:** hot_halo_cold_mode_density_ciso_corer_vfrac

**Description:** Returns the radius (in Mpc) of the core radius in a cored isothermal cold mode hot halo density profile. Assumes that the radius is a fixed fraction of the halo virial radius.

**Code lines:** 13

**Contained by:** module hot_halo_cold_mode_density_ciso_corer_virial_fraction
19.1. Program units

*Modules used:* dark_matter_halo_scales  galacticus_nodes

**file:** hot_halo.mass_distribution.F90  
*Description:* Contains a module which provides an object that implements hot halo mass distributions.  
*Code lines:* 827

**module:** hot_halo_mass_distributions  
*Description:* Provides an object that implements hot halo density profiles. An implementation of the hot halo mass distribution class for $\beta$-profile distributions. A null implementation of the hot halo mass distribution class. An implementation of the hot halo mass distribution class which uses the model of Ricotti and Shull [2000].  
*Code lines:* 807  
*Contained by:* file hot_halo.mass_distribution.F90  
*Modules used:* galacticus_nodes  iso_varying_string  mass_distributions

*Used by:*  
- function cooling_radius_isothermal  
- function cooling_rate_cole2000  
- function cooling_specific-angular-_momentum_constant_rotation  
- function galactic_structure_enclosed-mass  
- function galactic_structure_rotation-curve  
- function hot_halo_ram_pressure_force-_font2008_get  
- function hot_halo_ram_pressure-_timescale_ram_pressure_accel  
- function node_component_hot_halo-_standard_outflow_stripped_fraction

*function:* betaprobeconstructor  
*Description:* Default constructor for the betaProfile hot halo mass distribution class.  
*Code lines:* 9  
*Contained by:* module hot_halo_mass_distributions  
*Modules used:* input_parameters

*function:* betaprobeconstructor  
*Description:* Default constructor for the betaProfile hot halo mass distribution class.  
*Code lines:* 31  
*Contained by:* module hot_halo_mass_distributions  
*Modules used:* array_utilities  galacticus_error  input_parameters

*function:* betaprobeconstructor  
*Description:* Return the density in a single-betaProfile hot halo mass distribution.  
*Code lines:* 13  
*Contained by:* module hot_halo_mass_distributions  
*Modules used:* coordinates
19. Source Code Documentation

**function:** betaprofiledensitylogslope

*Description:* Return the logarithmic slope of the density of the hot halo at the given radius.

*Code lines:* 13

*Contained by:* module hot_halo_mass_distributions

*Modules used:* coordinates

**subroutine:** betaprofiledestructor

*Description:* Destructor for the betaProfile hot halo mass distribution class.

*Code lines:* 7

*Contained by:* module hot_halo_mass_distributions

**function:** betaprofileenclosedmass

*Description:* Return the mass enclosed in the hot halo at the given radius.

*Code lines:* 16

*Contained by:* module hot_halo_mass_distributions

**subroutine:** betaprofileinitialize

*Description:* Initialize the β-profile hot halo density profile for the given node.

*Code lines:* 22

*Contained by:* module hot_halo_mass_distributions

*Modules used:* hot_halo_mass_distributions_core_radii

**function:** betaprofileradialmoment

*Description:* Return the radial moment of the density profile of the hot halo to the given radius.

*Code lines:* 12

*Contained by:* module hot_halo_mass_distributions

**function:** betaprofilerotationnormalization

*Description:* Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for node. Specifically, the normalization, $A$, returned is such that $V_{rot} = AJ/M$.

*Code lines:* 12

*Contained by:* module hot_halo_mass_distributions

**function:** hot_halo_mass_distribution_density_task

*Description:* Computes the density at a given position for a dark matter profile.

*Code lines:* 18

*Contained by:* module hot_halo_mass_distributions

*Modules used:* galactic_structure_options

**function:** hot_halo_mass_distribution_enclosed_mass_task

*Description:* Computes the mass within a given radius for a dark matter profile.

*Code lines:* 19

*Contained by:* module hot_halo_mass_distributions

*Modules used:* galactic_structure_options
function: hot_halo_mass_distribution_rotation_curve_gradient_task
Description: Computes the rotation curve gradient at a given radius for the hot halo density profile.
Code lines: 26
Contained by: module hot_halo_mass_distributions
Modules used: galactic_structure_options numerical_constants_math numerical_constants_physical

function: hot_halo_mass_distribution_rotation_curve_task
Description: Computes the rotation curve at a given radius for the hot halo density profile.
Code lines: 20
Contained by: module hot_halo_mass_distributions
Modules used: galactic_structure_options numerical_constants_physical

interface: hothalomassdistribution
Code lines: 3
Contained by: module hot_halo_mass_distributions

interface: hothalomassdistributionbetaprofile
Description: Constructors for the $\beta$-profile hot halo mass distribution class.
Code lines: 4
Contained by: module hot_halo_mass_distributions

type: hothalomassdistributionclass
Code lines: 49
Contained by: module hot_halo_mass_distributions

function: hothalomassdistributionconstructordefault
Description: Return a pointer to the default hotHaloMassDistribution object.
Code lines: 8
Contained by: module hot_halo_mass_distributions

function: hothalomassdistributionconstructornamed
Description: Return a pointer to a newly created hotHaloMassDistribution object of the specified type.
Code lines: 37
Contained by: module hot_halo_mass_distributions
Modules used: galacticus_error iso_varying_string

function: hothalomassdistributiondensitylogslopenull
Description: Return the logarithmic slope of the density of the hot halo at the given radius.
Code lines: 9
Contained by: module hot_halo_mass_distributions
Modules used: galacticus_error

function: hothalomassdistributiondensitynull
Description: Return the density of the hot halo at the given radius.
Code lines: 9
Contained by: module hot_halo_mass_distributions
Modules used: galacticus_error
function: hothalomassdistributionenclosedmassnull
  Description: Return the mass enclosed in the hot halo at the given radius.
  Code lines: 9
  Contained by: module hot_halo_mass_distributions
  Modules used: galacticus_error

subroutine: hothalomassdistributioninitialize
  Description: Initialize the default hotHaloMassDistribution object.
  Code lines: 54
  Contained by: module hot_halo_mass_distributions
  Modules used: galacticus_error, input_parameters

function: hothalomassdistributionisfinalizable
  Description: Return true if this object can be finalized.
  Code lines: 7
  Contained by: module hot_halo_mass_distributions

interface: hothalomassdistributionnull
  Description: Constructors for the null hot halo mass distribution class.
  Code lines: 3
  Contained by: module hot_halo_mass_distributions

function: hothalomassdistributionradialmomentnull
  Description: Return the density of the hot halo at the given radius.
  Code lines: 9
  Contained by: module hot_halo_mass_distributions
  Modules used: galacticus_error

interface: hothalomassdistributionricotti2000
  Description: Constructors for the ricotti2000 hot halo mass distribution class.
  Code lines: 3
  Contained by: module hot_halo_mass_distributions

function: hothalomassdistributionrotationnormalizationnull
  Description: Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for node. Specifically, the normalization, A, returned is such that $V_{rot} = AJ/M$.
  Code lines: 8
  Contained by: module hot_halo_mass_distributions
  Modules used: galacticus_error

function: nulldefaultconstructor
  Description: Default constructor for the null hot halo mass distribution class.
  Code lines: 5
  Contained by: module hot_halo_mass_distributions

function: nulldensity
  Description: Return the density in a null hot halo mass distribution.
19.1. Program units

Code lines: 9
Contained by: module hot_halo_mass_distributions

function: nulldensitylogslope
Description: Return the logarithmic slope of the density of the hot halo at the given radius.
Code lines: 9
Contained by: module hot_halo_mass_distributions

subroutine: nulldestructor
Description: Destructor for the null hot halo mass distribution class.
Code lines: 7
Contained by: module hot_halo_mass_distributions

function: nullenclosedmass
Description: Return the mass enclosed in the hot halo at the given radius.
Code lines: 9
Contained by: module hot_halo_mass_distributions

function: nullradialmoment
Description: Return the density of the hot halo at the given radius.
Code lines: 9
Contained by: module hot_halo_mass_distributions

function: nullrotationnormalization
Description: Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for node. Specifically, the normalization, A, returned is such that \( V_{\text{rot}} = AJ/M \).
Code lines: 8
Contained by: module hot_halo_mass_distributions

function: ricotti2000defaultconstructor
Description: Default constructor for the ricotti2000 hot halo mass distribution class.
Code lines: 19
Contained by: module hot_halo_mass_distributions
Modules used: array_utilities galacticus_error

subroutine: ricotti2000initialize
Description: Initialize the ricotti2000 hot halo density profile for the given node. Parameterizations of \( \beta \) and core radius are taken from section 2.1 of Ricotti and Shull [2000].
Code lines: 33
Contained by: module hot_halo_mass_distributions
Modules used: dark_matter_halo_scales

file: hot_halo_mass_distribution.Ricotti2000.F90
Description: An implementation of the hot halo mass distribution class which uses the model of Ricotti and Shull [2000].
Code lines: 93

interface: hothalomassdistributionricotti2000
Description: Constructors for the ricotti2000 hot halo mass distribution class.
19. Source Code Documentation

**Code lines:** 3  
**Contained by:** file `hot_halo.mass_distribution.Ricotti2000.F90`

**function:** ricotti2000defaultconstructor  
**Description:** Default constructor for the ricotti2000 hot halo mass distribution class.  
**Code lines:** 19  
**Contained by:** file `hot_halo.mass_distribution.Ricotti2000.F90`  
**Modules used:** `array_utilities` `galacticus_error`

**subroutine:** ricotti2000initialize  
**Description:** Initialize the ricotti2000 hot halo density profile for the given node. Parameterizations of $\beta$ and core radius are taken from section 2.1 of Ricotti and Shull [2000].  
**Code lines:** 33  
**Contained by:** file `hot_halo.mass_distribution.Ricotti2000.F90`  
**Modules used:** `dark_matter_halo_scales`

**file:** `hot_halo.mass_distribution.beta_profile.F90`  
**Description:** An implementation of the hot halo mass distribution class for $\beta$-profile distributions.  
**Code lines:** 209

**function:** betaprofileconstructor  
**Description:** Default constructor for the betaProfile hot halo mass distribution class.  
**Code lines:** 9  
**Contained by:** file `hot_halo.mass_distribution.beta_profile.F90`  
**Modules used:** `input_parameters`

**function:** betaprofiledefaultconstructor  
**Description:** Default constructor for the betaProfile hot halo mass distribution class.  
**Code lines:** 31  
**Contained by:** file `hot_halo.mass_distribution.beta_profile.F90`  
**Modules used:** `array_utilities` `galacticus_error` `input_parameters`

**function:** betaprofiledensity  
**Description:** Return the density in a single-betaProfile hot halo mass distribution.  
**Code lines:** 13  
**Contained by:** file `hot_halo.mass_distribution.beta_profile.F90`  
**Modules used:** `coordinates`

**function:** betaprofiledensitylogslope  
**Description:** Return the logarithmic slope of the density of the hot halo at the given radius.  
**Code lines:** 13  
**Contained by:** file `hot_halo.mass_distribution.beta_profile.F90`  
**Modules used:** `coordinates`

**subroutine:** betaprofiledestructor  
**Description:** Destructor for the betaProfile hot halo mass distribution class.  
**Code lines:** 7  
**Contained by:** file `hot_halo.mass_distribution.beta_profile.F90`
function: betaprofileenclosedmass
Description: Return the mass enclosed in the hot halo at the given radius.
Code lines: 16
Contained by: file hot_halo.mass_distribution.beta_profile.F90

subroutine: betaprofileinitialize
Description: Initialize the $\beta$-profile hot halo density profile for the given node.
Code lines: 22
Contained by: file hot_halo.mass_distribution.beta_profile.F90
Modules used: hot_halo_mass_distributions_core_-radii

function: betaprofileradialmoment
Description: Return the radial moment of the density profile of the hot halo to the given radius.
Code lines: 12
Contained by: file hot_halo.mass_distribution.beta_profile.F90

function: betaprofilerotationnormalization
Description: Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for node. Specifically, the normalization, $A$, returned is such that $V_{\text{rot}} = AJ/M$.
Code lines: 12
Contained by: file hot_halo.mass_distribution.beta_profile.F90

interface: hothalomassdistributionbetaprofile
Description: Constructors for the $\beta$-profile hot halo mass distribution class.
Code lines: 4
Contained by: file hot_halo.mass_distribution.beta_profile.F90

file: hot_halo.mass_distribution.cored.core_radius.F90
Description: Contains a module which provides an object that implements core radii for cored hot halo mass distributions.
Code lines: 595

module: hot_halo_mass_distributions_core_radii
Description: Provides an object that implements core radii for hot halo mass distributions. An implementation of the hot halo mass distribution core radius class which sets the core radius to a fraction of the virial radius. An implementation of the hot halo mass distribution core radius class in which the core grows as the hot halo content is depleted.
Code lines: 575
Contained by: file hot_halo.mass_distribution.cored.core_radius.F90
Modules used: fgsl iso_varying_string galacticus_nodes tables
Used by: subroutine galacticus_state_retrieve subroutine galacticus_state_store subroutine betaprofileinitialize subroutine betaprofileinitialize

function: growingconstructor
Description: Default constructor for the growing hot halo mass distribution core radius class.
19. Source Code Documentation

**Code lines:** 14  
**Contained by:** module hot_halo_mass_distributions_core_radii  
**Modules used:** galacticus_error  

**function: growingcorevirialdensityfunction**  
**Description:** Returns the function \((1 + r_c^2)[1 - r_c \tan^{-1}(1/r_c)]\) which is proportional to the density at the virial radius of a cored isothermal profile with core radius \(r_c\) (in units of the virial radius) per unit mass.

**Code lines:** 8  
**Contained by:** module hot_halo_mass_distributions_core_radii

**function: growingdefaultconstructor**  
**Description:** Default constructor for the growing hot halo mass distribution core radius class.

**Code lines:** 36  
**Contained by:** module hot_halo_mass_distributions_core_radii  
**Modules used:** input_parameters

**subroutine: growingdestructor**  
**Description:** Destructor for the growing hot halo mass distribution class.

**Code lines:** 7  
**Contained by:** module hot_halo_mass_distributions_core_radii

**function: growingradius**  
**Description:** Return the core radius of the hot halo mass distribution.

**Code lines:** 63  
**Contained by:** module hot_halo_mass_distributions_core_radii  
**Modules used:** cosmology_parameters dark_matter_halo_scales

**subroutine: growingstaterestore**  
**Description:** Retrieve the tabulation state from the file.

**Code lines:** 13  
**Contained by:** module hot_halo_mass_distributions_core_radii  
**Modules used:** fgsl

**subroutine: growingstatestore**  
**Description:** Write the tabulation state to file.

**Code lines:** 10  
**Contained by:** module hot_halo_mass_distributions_core_radii  
**Modules used:** fgsl

**interface: hothalomassdistributioncoreradius**  
**Code lines:** 3  
**Contained by:** module hot_halo_mass_distributions_core_radii

**type: hothalomassdistributioncoreradiusclass**  
**Code lines:** 35  
**Contained by:** module hot_halo_mass_distributions_core_radii

**function: hothalomassdistributioncoreradiusconstructordefault**
19.1. Program units

**Description:** Return a pointer to the default `hotHaloMassDistributionCoreRadius` object.

**Code lines:** 8

**Contained by:** module `hot_halo_mass_distributions_core_radii`

**function:** `hothalomassdistributioncoreradiusconstructornamed`

**Description:** Return a pointer to a newly created `hotHaloMassDistributionCoreRadius` object of the specified type.

**Code lines:** 30

**Contained by:** module `hot_halo_mass_distributions_core_radii`

**Modules used:** `galacticus_error`, `iso_varying_string`

**subroutine:** `hothalomassdistributioncoreradiussdostateretrieve`

**Description:** Retrieve the state from file.

**Code lines:** 10

**Contained by:** module `hot_halo_mass_distributions_core_radii`

**subroutine:** `hothalomassdistributioncoreradiussdostatestore`

**Description:** Store the state to file.

**Code lines:** 10

**Contained by:** module `hot_halo_mass_distributions_core_radii`

**interface:** `hothalomassdistributioncoreradiussgrowing`

**Description:** Constructors for the growing hot halo mass distribution core radius class.

**Code lines:** 4

**Contained by:** module `hot_halo_mass_distributions_core_radii`

**subroutine:** `hothalomassdistributioncoreradiusinitialize`

**Description:** Initialize the default `hotHaloMassDistributionCoreRadius` object.

**Code lines:** 47

**Contained by:** module `hot_halo_mass_distributions_core_radii`

**Modules used:** `galacticus_error`, `input_parameters`, `iso_varying_string`

**function:** `hothalomassdistributioncoreradiusisfinalizable`

**Description:** Return true if this object can be finalized.

**Code lines:** 7

**Contained by:** module `hot_halo_mass_distributions_core_radii`

**function:** `hothalomassdistributioncoreradiusradiusnull`

**Description:** Return the core radius of the hot halo mass distribution.

**Code lines:** 8

**Contained by:** module `hot_halo_mass_distributions_core_radii`

**Modules used:** `galacticus_error`

**subroutine:** `hothalomassdistributioncoreradiussdostaterestore`

**Description:** Restore the state of the object to file.

**Code lines:** 9

**Contained by:** module `hot_halo_mass_distributions_core_radii`

**Modules used:** `fgsl`
subroutine: hothalamassdistributioncoreradiusstatestore
Description: Store the state of the object to file.
Code lines: 9
Contained by: module hot_halo_mass_distributions_core_radii
Modules used: fgsa1

interface: hothalamassdistributioncoreradiusvirialfraction
Description: Constructors for the virialFraction hot halo mass distribution core radius class.
Code lines: 4
Contained by: module hot_halo_mass_distributions_core_radii

function: virialfractionconstructor
Description: Default constructor for the virialFraction hot halo mass distribution core radius class.
Code lines: 9
Contained by: module hot_halo_mass_distributions_core_radii
Modules used: input_parameters

function: virialfractiondefaultconstructor
Description: Default constructor for the virialFraction hot halo mass distribution core radius class.
Code lines: 27
Contained by: module hot_halo_mass_distributions_core_radii
Modules used: input_parameters

subroutine: virialfractiondestructor
Description: Destructor for the virialFraction hot halo mass distribution class.
Code lines: 7
Contained by: module hot_halo_mass_distributions_core_radii

function: virialfractionradius
Description: Return the core radius of the hot halo mass distribution.
Code lines: 11
Contained by: module hot_halo_mass_distributions_core_radii
Modules used: dark_matter_halo_scales

file: hot_halo.mass_distribution.cored.core_radius.growing.F90
Description: An implementation of the hot halo mass distribution core radius class in which the core grows as the hot halo content is depleted.
Code lines: 217
Modules used: tables

function: growingconstructor
Description: Default constructor for the growing hot halo mass distribution core radius class.
Code lines: 14
Contained by: file hot_halo.mass_distribution.cored.core_radius.growing.F90
Modules used: galacticus_error, input_parameters

function: growingcorevirialdensityfunction
Description: Returns the function \((1 + r_c^2)[1 - r_c \tan^{-1}(1/r_c)]\) which is proportional to the density at the virial radius of a cored isothermal profile with core radius \(r_c\) (in units of the virial radius) per unit mass.
Code lines: 8
19.1. Program units

Contained by: file hot_halo.mass_distribution.cored.core_radius.growing.F90

function: growingdefaultconstructor
Description: Default constructor for the growing hot halo mass distribution core radius class.
Code lines: 36
Contained by: file hot_halo.mass_distribution.cored.core_radius.growing.F90
Modules used: input_parameters

subroutine: growingdestructor
Description: Destructor for the growing hot halo mass distribution class.
Code lines: 7
Contained by: file hot_halo.mass_distribution.cored.core_radius.growing.F90

function: growingradius
Description: Return the core radius of the hot halo mass distribution.
Code lines: 63
Contained by: file hot_halo.mass_distribution.cored.core_radius.growing.F90
Modules used: cosmology_parameters dark_matter_halo_scales

subroutine: growingstaterestore
Description: Retrieve the tabulation state from the file.
Code lines: 13
Contained by: file hot_halo.mass_distribution.cored.core_radius.growing.F90
Modules used: fgsl

subroutine: growingstatestore
Description: Write the tablulation state to file.
Code lines: 10
Contained by: file hot_halo.mass_distribution.cored.core_radius.growing.F90
Modules used: fgsl

interface: hothalomassdistributioncoreradiusgrowing
Description: Constructors for the growing hot halo mass distribution core radius class.
Code lines: 4
Contained by: file hot_halo.mass_distribution.cored.core_radius.growing.F90

file: hot_halo.mass_distribution.cored.core_radius.virial_radius_fraction.F90
Description: An implementation of the hot halo mass distribution core radius class which sets the core radius to a fraction of the virial radius.
Code lines: 103

interface: hothalomassdistributioncoreradiusvirialfraction
Description: Constructors for the virialFraction hot halo mass distribution core radius class.
Code lines: 4
Contained by: file hot_halo.mass_distribution.cored.core_radius.virial_radius_fraction.F90

function: virialfractionconstructor
Description: Default constructor for the virialFraction hot halo mass distribution core radius class.
Code lines: 9
19. Source Code Documentation

Contained by: file `hot_halo.mass_distribution.cored.core_radius.virial_radius_fraction.F90`
Modules used: `input_parameters`

**function: virialfractiondefaultconstructor**
*Description:* Default constructor for the `virialFraction` hot halo mass distribution core radius class.
*Code lines:* 27
*Contained by:* file `hot_halo.mass_distribution.cored.core_radius.virial_radius_fraction.F90`
*Modules used:* `input_parameters`

**subroutine: virialfractiondestructor**
*Description:* Destructor for the `virialFraction` hot halo mass distribution class.
*Code lines:* 7
*Contained by:* file `hot_halo.mass_distribution.cored.core_radius.virial_radius_fraction.F90`

**function: virialfractionradius**
*Description:* Return the core radius of the hot halo mass distribution.
*Code lines:* 11
*Contained by:* file `hot_halo.mass_distribution.cored.core_radius.virial_radius_fraction.F90`
*Modules used:* `dark_matter_halo_scales`

**file: hot_halo.mass_distribution.null.F90**
*Description:* A null implementation of the hot halo mass distribution class.
*Code lines:* 111

**interface: hothalomassdistributionnull**
*Description:* Constructors for the null hot halo mass distribution class.
*Code lines:* 3
*Contained by:* file `hot_halo.mass_distribution.null.F90`

**function: nulldefaultconstructor**
*Description:* Default constructor for the null hot halo mass distribution class.
*Code lines:* 5
*Contained by:* file `hot_halo.mass_distribution.null.F90`

**function: nulldensity**
*Description:* Return the density in a null hot halo mass distribution.
*Code lines:* 9
*Contained by:* file `hot_halo.mass_distribution.null.F90`

**function: nulldensitylogslope**
*Description:* Return the logarithmic slope of the density of the hot halo at the given radius.
*Code lines:* 9
*Contained by:* file `hot_halo.mass_distribution.null.F90`

**subroutine: nulldestructor**
*Description:* Destructor for the null hot halo mass distribution class.
*Code lines:* 7
*Contained by:* file `hot_halo.mass_distribution.null.F90`

**function: nullenclosedmass**
19.1. Program units

**Description:** Return the mass enclosed in the hot halo at the given radius.

**Code lines:** 9

**Contained by:** file `hot_halo.mass_distribution.null.F90`

**function:** nullradialmoment

**Description:** Return the density of the hot halo at the given radius.

**Code lines:** 9

**Contained by:** file `hot_halo.mass_distribution.null.F90`

**function:** nullrotationnormalization

**Description:** Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for node. Specifically, the normalization, $A$, returned is such that $V_{\text{rot}} = A J / M$.

**Code lines:** 8

**Contained by:** file `hot_halo.mass_distribution.null.F90`

**file:** `hot_halo.ram_pressure_force.F90`

**Description:** Contains a module that implements calculations of ram pressure stripping of hot halos.

**Code lines:** 100

**module:** `hot_halo.ram_pressure_forces`

**Description:** Implements calculations of ram pressure force

**Code lines:** 80

**Contained by:** file `hot_halo.ram_pressure_force.F90`

**Modules used:** `iso_varying_string`

**Used by:** function `hot_halo_ram_pressure_force`  function `hot_halo_ram_pressure_force_font2008_get`  function `hot_halo_ram_pressure_force_null`

**file:** `hot_halo.ram_pressure_force.Font2008.F90`

**Description:** Contains a module which implements a model of ram pressure stripping of hot halos based on the methods of Font et al. [2008].

**Code lines:** 78

**module:** `hot_halo.ram_pressure_force.Font2008`

**Description:** Implements a module which implements the calculation of the ram pressure force on hot halos based on the methods of Font et al. [2008].

**Code lines:** 57

**Contained by:** file `hot_halo.ram_pressure_force.Font2008.F90`

**Modules used:** `galacticus_nodes`

**Used by:** function `hot_halo_ram_pressure_force`
function: hot_halo_ram_pressure_force_font2008_get
    Description: Computes the hot halo ram pressure force
    Code lines: 29
    Contained by: module hot_halo_ram_pressure_force_font2008
    Modules used: hot_halo_mass_distributions kepler_orbits satellite_orbits

subroutine: hot_halo_ram_pressure_force_font2008_initialize
    Description: Initializes the “Font2008” hot halo ram pressure stripping module.
    Code lines: 11
    Contained by: module hot_halo_ram_pressure_force_font2008
    Modules used: iso_varying_string

file: hot_halo.ram_pressure_force.null.F90
    Description: Contains a module which implements a null ram pressure force for hot halos.
    Code lines: 52

module: hot_halo_ram_pressure_force_null
    Description: Implements a null ram pressure force for hot halos.
    Code lines: 32
    Contained by: file hot_halo.ram_pressure_force.null.F90
    Used by: function hot_halo_ram_pressure_force

function: hot_halo_ram_pressure_force_null_get
    Description: Computes the ram pressure force from the hot halo in the null implementation. Always returns zero.
    Code lines: 8
    Contained by: module hot_halo_ram_pressure_force_null
    Modules used: galacticus_nodes

subroutine: hot_halo_ram_pressure_force_null_initialize
    Description: Initializes the “Null” hot halo ram pressure stripping module.
    Code lines: 9
    Contained by: module hot_halo_ram_pressure_force_null
    Modules used: iso_varying_string

file: hot_halo.ram_pressure_stripping.F90
    Description: Contains a module that implements calculations of ram pressure stripping of hot halos.
    Code lines: 100

module: hot_halo_ram_pressure_stripping
    Description: Implements calculations of ram pressure stripping of hot halos.
    Code lines: 80
    Contained by: file hot_halo.ram_pressure_stripping.F90
    Modules used: iso_varying_string
    Used by: subroutine node_component_hot_halo_-cold_mode_rate_compute
             function node_component_hot_halo_-standard_outer_radius_growth_rate
function: hot_halo_ram_pressure_stripping_radius
Description: Return the ram pressure stripping radius for the hot halo of thisNode (in units of Mpc).
Code lines: 60
Contained by: module hot_halo_ram_pressure_stripping
Modules used: galacticus_error
galacticus_nodes
hot_halo_ram_pressure_stripping_font2008
virial_radii
input_parameters

file: hot_halo.ram_pressure_stripping.Font2008.F90
Description: Contains a module which implements a model of ram pressure stripping of hot halos based on the methods of Font et al. [2008].
Code lines: 149

module: hot_halo_ram_pressure_stripping_font2008
Description: Implements a module which implements a model of ram pressure stripping of hot halos based on the methods of Font et al. [2008].
Code lines: 128
Contained by: file hot_halo.ram_pressure_stripping.Font2008.F90
Modules used: galacticus_nodes
Used by: function hot_halo_ram_pressure_stripping_font2008_get

function: hot_halo_ram_pressure_stripping_font2008_get
Description: Computes the hot halo ram pressure stripping radius, assuming a null calculation in which that radius always equals the virial radius.
Code lines: 54
Contained by: module hot_halo_ram_pressure_stripping_font2008
Modules used: dark_matter_halo_scales
hot_halo_ram_pressure_forces
root_finder

subroutine: hot_halo_ram_pressure_stripping_font2008_initialize
Description: Initializes the “Font2008” hot halo ram pressure stripping module.
Code lines: 24
Contained by: module hot_halo_ram_pressure_stripping_font2008
Modules used: input_parameters
iso_varying_string

function: hot_halo_ram_pressure_stripping_radius_solver
Description: Root function used in finding the ram pressure stripping radius.
Code lines: 22
Contained by: module hot_halo_ram_pressure_stripping_font2008
Modules used: galactic_structure_enclosed_masses
galactic_structure_options
hot_halo_mass_distributions
numerical_constants_physical

file: hot_halo.ram_pressure_stripping.timescale.F90
Description: Contains a module that implements calculations of ram pressure stripping timescales for hot halos.
Code lines: 100
module: hot_halo_ram_pressure_stripping_timescales
Description: Implements calculations of ram pressure stripping timescales for hot halos.
Code lines: 80
Contained by: file hot_halo.ram_pressure_stripping.timescale.F90
Modules used: iso_varying_string
Used by: function node_component_hot_halo_standard_outer_radius_growth_rate subroutine node_component_hot_halo_standard_rate_compute

function: hot_halo_ram_pressure_stripping_timescale
Description: Return the ram pressure stripping radius for the hot halo of thisNode (in units of Mpc).
Code lines: 60
Contained by: module hot_halo_ram_pressure_stripping_timescales
Modules used: galacticus_error galacticus_nodes
hot_halo_ram_pressure_timescales_halo_dyntime hot_halo_ram_pressure_timescales_halo_dyntime ram_pressure_accel
input_parameters

file: hot_halo.ram_pressure_stripping.timescale.halo_dynamical_time.F90
Description: Contains a module which implements a calculation of hot halo ram pressure timescales based on the halo dynamical time.
Code lines: 55

module: hot_halo_ram_pressure_timescales_halo_dyntime
Description: Implements a calculation of hot halo ram pressure timescales based on the halo dynamical time.
Code lines: 35
Contained by: file hot_halo.ram_pressure_stripping.timescale.halo_dynamical_time.F90
Used by: function hot_halo_ram_pressure_stripping_timescale

function: hot_halo_ram_pressure_timescales_halo_dyntime
Description: Computes the hot halo ram pressure stripping timescale, assuming that it equals the halo dynamical time.
Code lines: 11
Contained by: module hot_halo_ram_pressure_timescales_halo_dyntime
Modules used: dark_matter_halo_scales galacticus_nodes

subroutine: hot_halo_ram_pressure_timescales_halo_dyntime_initialize
Description: Initializes the “halo dynamical time” hot halo ram pressure stripping timescale module.
Code lines: 9
Contained by: module hot_halo_ram_pressure_timescales_halo_dyntime
Modules used: iso_varying_string

file: hot_halo.ram_pressure_stripping.timescale.ram_pressure_acceleration.F90
Description: Contains a module which implements a calculation of hot halo ram pressure timescales based on an estimate of the acceleration due to ram pressure.
Code lines: 92

module: hot_halo_ram_pressure_timescales_ram_pressure_accel
19.1. Program units

Description: Implements a calculation of hot halo ram pressure timescales based on an estimate of the acceleration due to ram pressure.

Code lines: 71

Contained by: file hot_halo.ram_pressure_stripping.timescale.ram_pressure_acceleration.F90

Used by: function hot_halo_ram_pressure_stripping_timescale

**function: hot_halo_ram_pressure_timescale_ram_pressure_accel**

**Description:** Computes the hot halo ram pressure stripping timescale, based on the acceleration due to ram pressure forces. This timescale is approximated as \[ \tau \approx \sqrt{2} \frac{r_{\text{outer}} \Sigma_{\text{outer}}}{P_{\text{ram}}} \], where \( r_{\text{outer}} \) is the current outer radius of the hot halo, \( \Sigma_{\text{outer}} \) is the surface density at that radius, and \( P_{\text{ram}} \) is the ram pressure force (per unit area). The surface density is approximated as \( \Sigma_{\text{outer}} \approx r_{\text{outer}} \rho_{\text{outer}} \), where \( \rho_{\text{outer}} \) is the density at the outer radius.

Code lines: 47

Contained by: module hot_halo_ram_pressure_timescales_ram_pressure_accel

Modules used:
- dark_matter_halo_scales
- galacticus_nodes
- hot_halo_mass_distributions
- hot_halo_ram_pressure_forces
- numerical_constants_astronomical
- numerical_constants_physical
- numerical_constants_prefixes

**subroutine: hot_halo_ram_pressure_timescales_ram_pressure_accel_initialize**

**Description:** Initializes the “ram pressure acceleration” hot halo ram pressure stripping timescale module.

Code lines: 9

Contained by: module hot_halo_ram_pressure_timescales_ram_pressure_accel

Modules used:
- iso_varying_string

**file: hot_halo.ram_pressure_stripping.virial_radius.F90**

**Description:** Contains a module which implements a null hot halo ram pressure stripping calculation, by simply returning the virial radius as the ram pressure stripping radius.

Code lines: 58

**module: hot_halo_ram_pressure_stripping_virial_radii**

**Description:** Implements a null hot halo ram pressure stripping calculation, by simply returning the virial radius as the ram pressure stripping radius.

Code lines: 37

Contained by: file hot_halo.ram_pressure_stripping.virial_radius.F90

Used by: function hot_halo_ram_pressure_stripping_radius

**subroutine: hot_halo_ram_pressure_stripping_virial_radii_initialize**

**Description:** Initializes the “virial radius” hot halo ram pressure stripping module.

Code lines: 9

Contained by: module hot_halo_ram_pressure_stripping_virial_radii

Modules used:
- iso_varying_string

**function: hot_halo_ram_pressure_stripping_virial_radius**

**Description:** Computes the hot halo ram pressure stripping radius, assuming a null calculation in which that radius always equals the virial radius.

Code lines: 12
19. Source Code Documentation

**Contained by:** module `hot_halo_ram_pressure_stripping_virial_radii`

**Modules used:** `dark_matter_halo_scales` `galacticus_nodes`

**file:** `hot_halo.temperature_profile.F90`

**Description:** Contains a module that implements calculations of the hot halo gas temperature profile.

**Code lines:** 130

**module:** `hot_halo_temperature_profile`

**Code lines:** 110

**Contained by:** file `hot_halo.temperature_profile.F90`

**Modules used:** `galacticus_nodes` `hot_halo_temperature_profile_virial`

**Used by:**
- function `cooling_radius_isothermal`
- function `cooling_radius_growth_rate_simple`
- function `cooling_radius_root`
- subroutine `node_component_black_hole_standard_mass_accretion_rate`

**function:** `hot_halo_temperature`

**Description:** Return the temperature of the hot halo in `thisNode` at radius `radius`.

**Code lines:** 13

**Contained by:** module `hot_halo_temperature_profile`

**interface:** `hot_halo_temperature_get_template`

**Code lines:** 6

**Contained by:** module `hot_halo_temperature_profile`

**function:** `hot_halo_temperature_get_template`

**Code lines:** 4

**Contained by:** interface `hot_halo_temperature_get_template`

**subroutine:** `hot_halo_temperature_initialize`

**Description:** Initialize the hot halo temperature module.

**Code lines:** 41

**Contained by:** module `hot_halo_temperature_profile`

**Modules used:** `galacticus_error` `input_parameters`

**function:** `hot_halo_temperature_logarithmic_slope`

**Description:** Return the temperature of the hot halo in `thisNode` at radius `radius`.

**Code lines:** 13

**Contained by:** module `hot_halo_temperature_profile`

**file:** `hot_halo.temperature_profile.virial.F90`

**Description:** Contains a module which implements an isothermal (virial temperature) profile for hot gas halos.

**Code lines:** 72

**module:** `hot_halo_temperature_profile_virial`

**Description:** Implements an isothermal (virial temperature) profile for hot gas halos.

**Code lines:** 52
function: hot_halo_temperature_logarithmic_slope_virial_get
Description: Compute the logarithmic slope of the temperature at radius radius in an isothermal temperature profile for thisNode.
Code lines: 10
Contained by: module hot_halo_temperature_profile_virial
Modules used: galacticus_nodes

subroutine: hot_halo_temperature_virial
Description: Initialize the cored isothermal hot halo temperature profile module.
Code lines: 13
Contained by: module hot_halo_temperature_profile_virial
Modules used: iso_varying_string

function: hot_halo_temperature_virial_get
Description: Compute the temperature at radius radius in an isothermal (virial) temperature profile for thisNode.
Code lines: 12
Contained by: module hot_halo_temperature_profile_virial
Modules used: dark_matter_halo_scales galacticus_nodes

file: inc_gam.F90
Code lines: 1623

module: incomplete_gamma
Code lines: 1620
Contained by: file inc_gam.F90
Modules used: constants_nswc
Used by: function inverse_gamma_function_-_incomplete_complementary

function: derf
Code lines: 39
Contained by: module incomplete_gamma

function: derfc0
Code lines: 63
Contained by: module incomplete_gamma

function: derfc1
Code lines: 64
Contained by: module incomplete_gamma

function: derfi
Code lines: 97
Contained by: module incomplete_gamma

function: dgam1
19. Source Code Documentation

Code lines: 125
Contained by: module incomplete_gamma

function: dgamln
Code lines: 46
Contained by: module incomplete_gamma

function: dgamma
Code lines: 112
Contained by: module incomplete_gamma

function: dgm1n1
Code lines: 13
Contained by: module incomplete_gamma

function: dlnlrel
Code lines: 34
Contained by: module incomplete_gamma

function: dpdel
Code lines: 29
Contained by: module incomplete_gamma

subroutine: dpni
Code lines: 56
Contained by: module incomplete_gamma

function: drcomp
Code lines: 32
Contained by: module incomplete_gamma

function: drexp
Code lines: 40
Contained by: module incomplete_gamma

function: drlog
Code lines: 61
Contained by: module incomplete_gamma

function: dsin1
Code lines: 64
Contained by: module incomplete_gamma

subroutine: gaminv
Code lines: 326
Contained by: module incomplete_gamma

subroutine: gratio
Code lines: 345
19.1. Program units

Contained by: module `incomplete_gamma`

**file: instruments.filters.F90**
*Description:* Contains a module which implements calculations of filter response curves.
*Code lines:* 212

**module: instruments_filters**
*Description:* Implements calculations of filter response curves.
*Code lines:* 192
*Contained by:* file `instruments.filters.F90`
*Modules used:* `fgsl`, `iso_varying_string`
*Used by:* subroutine `stellar_luminosities_-initialize`
function `filter_luminosity_integrand`
function `filter_luminosity_integrand_ab`
function `stellar_population_luminosity`

**function: filter_extent**
*Description:* Return an array containing the minimum and maximum wavelengths tabulated for this specified filter.
*Code lines:* 11
*Contained by:* module `instruments_filters`

**function: filter_get_index**
*Description:* Return the index for the specified filter, loading that filter if necessary.
*Code lines:* 26
*Contained by:* module `instruments_filters`

**function: filter_name**
*Description:* Return the name of the specified filter.
*Code lines:* 8
*Contained by:* module `instruments_filters`

**function: filter_response**
*Description:* Return the filter response function at the given wavelength (specified in Angstroms). Note that we follow the convention of Hogg et al. [2002] and assume that the filter response gives the fraction of incident photons received by the detector at a given wavelength, multiplied by the relative photon response (which will be 1 for a photon-counting detector such as a CCD, or proportional to the photon energy for a bolometer/calorimeter type detector.
*Code lines:* 15
*Contained by:* module `instruments_filters`
*Modules used:* `numerical_interpolation`

**subroutine: filter_response_load**
*Description:* Load a filter response curve.
*Code lines:* 85
*Contained by:* module `instruments_filters`
*Modules used:* `fox_dom`, `galacticus_error`, `galacticus_input_paths`, `io_xml`, `memory_management`, `string_handling`
function: filter_vega_offset
Description: Return the Vega to AB magnitude offset for the specified filter.
Code lines: 9
Contained by: module instruments_filters
Modules used: galacticus_error

type: filtertype
Description: A structure which holds filter response curves.
Code lines: 11
Contained by: module instruments_filters

file: intergalactic_medium.state.F90
Description: Contains a module which provides a class for calculations of the intergalactic medium thermal and ionization state.
Code lines: 1577

module: intergalactic_medium_state
Description: Provides a class for calculations of the intergalactic medium thermal and ionization state. An implementation of the intergalactic medium state class for a simplistic model of instantaneous and full reionization. An implementation of the intergalactic medium state class for an internal model of instantaneous and full reionization. An implementation of the intergalactic medium state class in which state is read from file. An implementation of the intergalactic medium state class in which state is computed using RECFAST.
Code lines: 1557
Contained by: file intergalactic_medium.state.F90
Modules used: cosmology_functions

Used by: function coldmodechemicalmasses
function naozbarkana2007failedfraction
function naozbarkana2007failedfraction
function coldmodedefaultconstructor
function simpledefaultconstructor
module intergalactic_medium_state_-internal_evolver
module radiation_intergalactic_-background_internal

function: fileconstructor
Description: Constructor for the file IGM state class.
Code lines: 9
Contained by: module intergalactic_medium_state
Modules used: cosmology_functions

function: filedefaultconstructor
Description: Default constructor for the file IGM state class.
Code lines: 26
Contained by: module intergalactic_medium_state
Modules used: input_parameters

function: fileelectronfraction
Description: Return the electron fraction in the intergalactic medium at the specified time by interpolating in tabulated data,
19.1. Program units

- **Code lines:** 12
- **Contained by:** module `intergalactic_medium_state`
- **Modules used:** `numerical_interpolation`

### function: fileneutralheliumfraction
- **Description:** Return the neutral helium fraction in the intergalactic medium at the specified time by interpolating in tabulated data.
- **Code lines:** 12
- **Contained by:** module `intergalactic_medium_state`
- **Modules used:** `numerical_interpolation`

### function: fileneutralhydrogenfraction
- **Description:** Return the neutral hydrogen fraction in the intergalactic medium at the specified time by interpolating in tabulated data.
- **Code lines:** 12
- **Contained by:** module `intergalactic_medium_state`
- **Modules used:** `numerical_interpolation`

### subroutine: filereaddata
- **Description:** Read in data describing the state of the intergalactic medium.
- **Code lines:** 45
- **Contained by:** module `intergalactic_medium_state`
- **Modules used:** `cosmology_functions` `fox_dom` `galacticus_error` `io_xml`

### function: filesinglyionizedheliumfraction
- **Description:** Return the neutral helium fraction in the intergalactic medium at the specified time by interpolating in tabulated data.
- **Code lines:** 12
- **Contained by:** module `intergalactic_medium_state`
- **Modules used:** `numerical_interpolation`

### function: filetemperature
- **Description:** Return the temperature of the intergalactic medium at the specified time by interpolating in tabulated data.
- **Code lines:** 12
- **Contained by:** module `intergalactic_medium_state`
- **Modules used:** `numerical_interpolation`

### interface: intergalacticmediumstate
- **Code lines:** 3
- **Contained by:** module `intergalactic_medium_state`

### type: intergalacticmediumstateclass
- **Code lines:** 82
- **Contained by:** module `intergalactic_medium_state`

### function: intergalacticmediumstateconstructordefault
- **Description:** Return a pointer to the default `intergalacticMediumState` object.
- **Code lines:** 8
function: intergalacticmediumstateconstructornamed
Description: Return a pointer to a newly created intergalacticMediumState object of the specified type.
Code lines: 44
Contained by: module intergalactic_medium_state
Modules used: galacticus_error iso_varying_string

function: intergalacticmediumstatedoublyionizedheliumfraction
Description: Return the doubly-ionized fraction of helium in the IGM at the given time.
Code lines: 7
Contained by: module intergalactic_medium_state

function: intergalacticmediumstateelectronfractionnull
Description: Return the electron fraction (relative to hydrogen) in the IGM at the given time.
Code lines: 8
Contained by: module intergalactic_medium_state
Modules used: galacticus_error

function: intergalacticmediumstateelectronscatteringintegrand
Description: Integrand for electron scattering optical depth calculations.
Code lines: 27
Contained by: module intergalactic_medium_state
Modules used: cosmology_functions cosmology_parameters iso_c_binding numerical_constants_astronomical numerical_constants_physical

function: intergalacticmediumstateelectronscatteringopticaldepth
Description: Return the electron scattering optical depth from the present day back to the given time in the IGM.
Code lines: 24
Contained by: module intergalactic_medium_state
Modules used: galacticus_error

subroutine: intergalacticmediumstateelectronscatteringtabulate
Description: Construct a table of electron scattering optical depth as a function of cosmological time.
Code lines: 49
Contained by: module intergalactic_medium_state
Modules used: cosmology_functions fgsl iso_c_binding numerical_integration galacticus_error

function: intergalacticmediumstateelectronscatteringtime
Description: Return the cosmological time at which the given electron scattering opticalDepth is reached (integrating from the present day) in the IGM.
Code lines: 35
Contained by: module intergalactic_medium_state
Modules used: cosmology_functions galacticus_error

interface: intergalacticmediumstatefile
19.1. Program units

**Description:** Constructors for the file intergalactic medium state class.

**Code lines:** 4

**Contained by:** module `intergalactic_medium_state`

**Subroutine:** `intergalacticmediumstateinitialize`

**Description:** Initialize the default `intergalacticMediumState` object.

**Code lines:** 64

**Contained by:** module `intergalactic_medium_state`

**Modules used:** `galacticus_error`, `input_parameters`, `iso_varying_string`

**Interface:** `intergalacticmediumstateinternal`

**Description:** Constructors for the internal intergalactic medium state class.

**Code lines:** 3

**Contained by:** module `intergalactic_medium_state`

**Function:** `intergalacticmediumstateisfinalizable`

**Description:** Return true if this object can be finalized.

**Code lines:** 7

**Contained by:** module `intergalactic_medium_state`

**Function:** `intergalacticmediumstateneutralheliumfractionnull`

**Description:** Return the neutral fraction of helium in the IGM at the given time.

**Code lines:** 8

**Contained by:** module `intergalactic_medium_state`

**Modules used:** `galacticus_error`

**Function:** `intergalacticmediumstateneutralhydrogenfractionnull`

**Description:** Return the neutral fraction of hydrogen in the IGM at the given time.

**Code lines:** 8

**Contained by:** module `intergalactic_medium_state`

**Modules used:** `galacticus_error`

**Interface:** `intergalacticmediumstaterecfast`

**Description:** Constructors for the RECFAST intergalactic medium state class.

**Code lines:** 4

**Contained by:** module `intergalactic_medium_state`

**Interface:** `intergalacticmediumstatesimple`

**Description:** Constructors for the simple intergalactic medium state class.

**Code lines:** 4

**Contained by:** module `intergalactic_medium_state`

**Function:** `intergalacticmediumstatesinglyionizedheliumfractionnull`

**Description:** Return the singly-ionized fraction of helium in the IGM at the given time.

**Code lines:** 8

**Contained by:** module `intergalactic_medium_state`

**Modules used:** `galacticus_error`
function: intergalacticmediumstatesinglyionizedhydrogenfraction
Description: Return the singly-ionized fraction of hydrogen in the IGM at the given time.
Code lines: 7
Contained by: module intergalactic_medium_state

function: intergalacticmediumstatetemperaturenull
Description: Return the temperature (in Kelvin) of the IGM at the given time.
Code lines: 8
Contained by: module intergalactic_medium_state
Modules used: galacticus_error

function: internaldefaultconstructor
Description: Default constructor for the internal IGM state class.
Code lines: 23
Contained by: module intergalactic_medium_state
Modules used: input_parameters

subroutine: internaldensityhelium1set
Description: Set He1 densities in the internal intergalactic medium state class.
Code lines: 11
Contained by: module intergalactic_medium_state
Modules used: memory_management

subroutine: internaldensityhelium2set
Description: Set He2 densities in the internal intergalactic medium state class.
Code lines: 11
Contained by: module intergalactic_medium_state
Modules used: memory_management

subroutine: internaldensityhelium3set
Description: Set He3 densities in the internal intergalactic medium state class.
Code lines: 11
Contained by: module intergalactic_medium_state
Modules used: memory_management

subroutine: internaldensityhydrogen1set
Description: Set H1 densities in the internal intergalactic medium state class.
Code lines: 11
Contained by: module intergalactic_medium_state
Modules used: memory_management

subroutine: internaldensityhydrogen2set
Description: Set H2 densities in the internal intergalactic medium state class.
Code lines: 11
Contained by: module intergalactic_medium_state
Modules used: memory_management
function: internalelectronfraction
Description: Return the electron fraction of the IGM in the internal model.
Code lines: 37
Contained by: module intergalactic_medium_state
Modules used: fgsl
  iso_c_binding
  numerical_interpolation

function: internalfilteringmass
Description: Return the filtering mass of the IGM in the internal model.
Code lines: 31
Contained by: module intergalactic_medium_state
Modules used: fgsl
  iso_c_binding
  numerical_interpolation

subroutine: internalmassfilteringset
Description: Set filtering masses in the internal intergalactic medium state class.
Code lines: 11
Contained by: module intergalactic_medium_state
Modules used: memory_management

function: internalneutralheliumfraction
Description: Return the neutral helium fraction of the IGM in the internal model.
Code lines: 35
Contained by: module intergalactic_medium_state
Modules used: fgsl
  iso_c_binding
  numerical_interpolation

function: internalneutralhydrogenfraction
Description: Return the neutral hydrogen fraction of the IGM in the internal model.
Code lines: 35
Contained by: module intergalactic_medium_state
Modules used: fgsl
  iso_c_binding
  numerical_interpolation

function: internalsinglyionizedheliumfraction
Description: Return the singly ionized helium fraction of the IGM in the internal model.
Code lines: 35
Contained by: module intergalactic_medium_state
Modules used: fgsl
  iso_c_binding
  numerical_interpolation

function: internaltemperature
Description: Return the temperature of the IGM in the internal model.
Code lines: 31
Contained by: module intergalactic_medium_state
Modules used: fgsl
  iso_c_binding
  numerical_interpolation
subroutine: internaltemperatureset
Description: Set temperatures in the internal intergalatic medium state class.
Code lines: 11
Contained by: module intergalactic_medium_state
Modules used: memory_management

subroutine: internaltimeset
Description: Set times in the internal intergalatic medium state class.
Code lines: 11
Contained by: module intergalactic_medium_state
Modules used: memory_management

function: recfastconstructor
Description: Constructor for the RecFast IGM state class.
Code lines: 48
Contained by: module intergalactic_medium_state
Modules used: cosmology_parameters fox_wxml galacticus_input_paths input_parameters numerical_constants_astronomical system_command

function: recfastdefaultconstructor
Description: Default constructor for the RecFast IGM state class.
Code lines: 10
Contained by: module intergalactic_medium_state
Modules used: cosmology_parameters

function: simpleconstructor
Description: Constructor for the simple IGM state class.
Code lines: 13
Contained by: module intergalactic_medium_state
Modules used: cosmology_functions

function: simpledefaultconstructor
Description: Default constructor for the simple IGM state class.
Code lines: 50
Contained by: module intergalactic_medium_state
Modules used: input_parameters

function: simpleselectronfraction
Description: Return the electron fraction of the IGM in the simple model.
Code lines: 13
Contained by: module intergalactic_medium_state
Modules used: numerical_constants_astronomical

function: simpleneutralheliumfraction
Description: Return the neutral helium fraction of the IGM in the simple model.
Code lines: 13
Contained by: module intergalactic_medium_state
19.1. Program units

*Modules used:* numerical_constants_astronomical

**function:** simpleneutralhydrogenfraction
*Description:* Return the neutral hydrogen fraction of the IGM in the simple model.
*Code lines:* 13
*Contained by:* module intergalactic_medium_state
*Modules used:* numerical_constants_astronomical

**function:** simplesinglyionizedheliumfraction
*Description:* Return the singly-ionized helium fraction of the IGM in the simple model.
*Code lines:* 13
*Contained by:* module intergalactic_medium_state
*Modules used:* numerical_constants_astronomical

**function:** simpletemperature
*Description:* Return the temperature of the IGM in the simple model.
*Code lines:* 12
*Contained by:* module intergalactic_medium_state

**file:** intergalactic_medium.state.RecFast.F90
*Description:* An implementation of the intergalactic medium state class in which state is computed using RecFast.
*Code lines:* 99

**interface:** intergalacticmediumstaterecfast
*Description:* Constructors for the RecFast intergalactic medium state class.
*Code lines:* 4
*Contained by:* file intergalactic_medium.state.RecFast.F90

**function:** recfastconstructor
*Description:* Constructor for the RecFast IGM state class.
*Code lines:* 48
*Contained by:* file intergalactic_medium.state.RecFast.F90
*Modules used:* cosmology_parameters fox_wxml galacticus_input_paths input_parameters numerical_constants_astronomical system_command

**function:** recfastdefaultconstructor
*Description:* Default constructor for the RecFast IGM state class.
*Code lines:* 10
*Contained by:* file intergalactic_medium.state.RecFast.F90
*Modules used:* cosmology_parameters

**file:** intergalactic_medium.state.file.F90
*Description:* An implementation of the intergalactic medium state class in which state is read from file.
*Code lines:* 220
*Modules used:* fgsl

**function:** fileconstructor
**19. Source Code Documentation**

**Description:** Constructor for the file IGM state class.

**Code lines:** 9

**Contained by:** file `intergalactic_medium.state.file.F90`

**Modules used:** `cosmology_functions`

**function:** filedefaultconstructor

**Description:** Default constructor for the file IGM state class.

**Code lines:** 26

**Contained by:** file `intergalactic_medium.state.file.F90`

**Modules used:** `input_parameters`

**function:** fileelectronfraction

**Description:** Return the electron fraction in the intergalactic medium at the specified time by interpolating in tabulated data,

**Code lines:** 12

**Contained by:** file `intergalactic_medium.state.file.F90`

**Modules used:** `numerical_interpolation`

**function:** fileneutralheliumfraction

**Description:** Return the neutral helium fraction in the intergalactic medium at the specified time by interpolating in tabulated data,

**Code lines:** 12

**Contained by:** file `intergalactic_medium.state.file.F90`

**Modules used:** `numerical_interpolation`

**function:** fileneutralhydrogenfraction

**Description:** Return the neutral hydrogen fraction in the intergalactic medium at the specified time by interpolating in tabulated data,

**Code lines:** 12

**Contained by:** file `intergalactic_medium.state.file.F90`

**Modules used:** `numerical_interpolation`

**subroutine:** filereaddata

**Description:** Read in data describing the state of the intergalactic medium.

**Code lines:** 45

**Contained by:** file `intergalactic_medium.state.file.F90`

**Modules used:** `cosmology_functions`  
`fox_dom`  
`galacticus_error`  
`io_xml`

**function:** filesinglyionizedheliumfraction

**Description:** Return the neutral helium fraction in the intergalactic medium at the specified time by interpolating in tabulated data,

**Code lines:** 12

**Contained by:** file `intergalactic_medium.state.file.F90`

**Modules used:** `numerical_interpolation`

**function:** filetemperature

**Description:** Return the temperature of the intergalactic medium at the specified time by interpolating in tabulated data.

**Code lines:** 12

**Contained by:** file `intergalactic_medium.state.file.F90`
19.1. Program units

Modules used: numerical_interpolation

interface: intergalacticmediumstatefile
Description: Constructors for the file intergalactic medium state class.
Code lines: 4
Contained by: file intergalactic_medium.state.file.F90

file: intergalactic_medium.state.internal.F90
Description: An implementation of the intergalactic medium state class for an internal model of instantaneous and full reionization.
Code lines: 453

interface: intergalacticmediumstateinternal
Description: Constructors for the internal intergalactic medium state class.
Code lines: 3
Contained by: file intergalactic_medium.state.internal.F90

function: internaldefaultconstructor
Description: Default constructor for the internal IGM state class.
Code lines: 23
Contained by: file intergalactic_medium.state.internal.F90
Modules used: input_parameters

subroutine: internaldensityhelium1set
Description: Set He1 densities in the internal intergalactic medium state class.
Code lines: 11
Contained by: file intergalactic_medium.state.internal.F90
Modules used: memory_management

subroutine: internaldensityhelium2set
Description: Set He2 densities in the internal intergalactic medium state class.
Code lines: 11
Contained by: file intergalactic_medium.state.internal.F90
Modules used: memory_management

subroutine: internaldensityhelium3set
Description: Set He3 densities in the internal intergalactic medium state class.
Code lines: 11
Contained by: file intergalactic_medium.state.internal.F90
Modules used: memory_management

subroutine: internaldensityhydrogen1set
Description: Set H1 densities in the internal intergalactic medium state class.
Code lines: 11
Contained by: file intergalactic_medium.state.internal.F90
Modules used: memory_management

subroutine: internaldensityhydrogen2set
Description: Set H2 densities in the internal intergalactic medium state class.
19. Source Code Documentation

Code lines: 11
Contained by: file intergalactic_medium.state.internal.F90
Modules used: memory_management

function: internalelectronfraction
Description: Return the electron fraction of the IGM in the internal model.
Code lines: 37
Contained by: file intergalactic_medium.state.internal.F90
Modules used: fgsl isoc_binding numerical_interpolation

function: internalfilteringmass
Description: Return the filtering mass of the IGM in the internal model.
Code lines: 31
Contained by: file intergalactic_medium.state.internal.F90
Modules used: fgsl isoc_binding numerical_interpolation

subroutine: internalmassfilteringset
Description: Set filtering masses in the internal intergalactic medium state class.
Code lines: 11
Contained by: file intergalactic_medium.state.internal.F90
Modules used: memory_management

function: internalneutralheliumfraction
Description: Return the neutral helium fraction of the IGM in the internal model.
Code lines: 35
Contained by: file intergalactic_medium.state.internal.F90
Modules used: fgsl isoc_binding numerical_interpolation

function: internalneutralhydrogenfraction
Description: Return the neutral hydrogen fraction of the IGM in the internal model.
Code lines: 35
Contained by: file intergalactic_medium.state.internal.F90
Modules used: fgsl isoc_binding numerical_interpolation

function: internalsinglyionizedheliumfraction
Description: Return the singly ionized helium fraction of the IGM in the internal model.
Code lines: 35
Contained by: file intergalactic_medium.state.internal.F90
Modules used: fgsl isoc_binding numerical_interpolation

function: internaltemperature
Description: Return the temperature of the IGM in the internal model.
Code lines: 31
Contained by: file intergalactic_medium.state.internal.F90
19.1. Program units

---

**Modules used:**
- fgsl
- iso_c_binding
- numerical_interpolation

**subroutine: internaltemperatureset**

*Description:* Set temperatures in the internal intergalatic medium state class.

*Code lines:* 11

*Contained by:* file `intergalactic_medium.state.internal.F90`

*Modules used:* memory_management

**subroutine: internaltimeset**

*Description:* Set times in the internal intergalatic medium state class.

*Code lines:* 11

*Contained by:* file `intergalactic_medium.state.internal.F90`

*Modules used:* memory_management

**file: intergalactic_medium.state.internal.evolver.F90**

*Description:* Contains a module which evolves the temperature and ionization states of baryons in the IGM.

*Code lines:* 670

**module: intergalactic_medium_state_internal_evolver**

*Description:* Evolves the temperature and ionization states of baryons in the IGM.

*Code lines:* 648

*Contained by:* file `intergalactic_medium.state.internal.evolver.F90`

*Modules used:* abundances_structure, cosmology_functions, fgs1, intergalactic_medium_state

*Used by:* function `galacticus_task_evolve_tree`

**subroutine: intergalactic_medium_state_internal_initialize**

*Description:* Attach an initial event to a merger tree to cause the properties update function to be called.

*Code lines:* 184

*Contained by:* module `intergalactic_medium_state_internal_evolver`

*Modules used:* cosmology_functions, cosmology_parameters, galacticus_error, galacticus_nodes, galacticus_output_times, input_parameters, linear_growth, memory_management, numerical_comparison, numerical_constants_atomic, numerical_constants_units, numerical_constants_math, numerical_ranges, power_spectra

**function: intergalactic_medium_state_internal_odes**

*Description:* Evaluates the ODEs controlling the evolution temperature.

*Code lines:* 248

*Contained by:* module `intergalactic_medium_state_internal_evolver`

*Modules used:* atomic_cross_sections_ionization_photo, atomic_ionization_potentials

---

1195
19. Source Code Documentation

atomic_radiation_gaunt_factors | atomic_rates_excitation_collisonal
atomic_rates_ionization_collisonal | atomic_rates_recombination_dielectronic
atomic_rates_recombination_radiative | atomic_rates_recombination_radiative
cosmology_functions | cosmology_parameters
fgsl | iso_c_binding
linear_growth | numerical_constants_astronomical
numerical_constants_atomic | numerical_constants_math
numerical_constants_physical | numerical_constants_prefixes
numerical_constants_units | numerical_integration
ode_solver_error_codes | power_spectra
radiation_structure

**function:** photoionization_heating_rate_integrand

**Description:** Integrand function used to compute the rate of photoionization heating of an ionic species.

**Code lines:** 16

**Contained by:** function intergalactic_medium_state_internal_odes

**function:** photoionization_rate_integrand

**Description:** Integrand function used to compute the rate of photoionizations of an ionic species.

**Code lines:** 16

**Contained by:** function intergalactic_medium_state_internal_odes

**function:** intergalactic_medium_state_internal_update

**Description:** Update the properties for a given universe.

**Code lines:** 166

**Contained by:** module intergalactic_medium_state_internal_evolver

**Modules used:**
- arrays_search
- cosmology_functions
- cosmology_parameters
- fodeiv2
- galactic_structure_options
- galacticus_display
- galacticus_error
- galacticus_hdf5
- galacticus_nodes
- galacticus_output_times
- iso_hdf5
- iso_varying_string
- linear_growth
- numerical_constants_astronomical
- numerical_constants_math
- numerical_constants_prefixes
- numerical_constants_physical
- numerical_constants_units
- numerical_integration
- odeiv2_solver
- power_spectra
- star_formation_imf
- stellar_population_spectra

**function:** rlss

**Description:** Evaluate the $r_{LSS}$ parameter of Naoz and Barkana [2007] using their fitting formula.

**Code lines:** 13

**Contained by:** module intergalactic_medium_state_internal_evolver

**file:** intergalactic_medium_state.simple.F90

**Description:** An implementation of the intergalactic medium state class for a simplistic model of instantaneous and full reionization.

**Code lines:** 187
interface: intergalacticmediumstatesimple
  Description: Constructors for the simple intergalactic medium state class.
  Code lines: 4
  Contained by: file intergalactic_medium.state.simple.F90

function: simpleconstructor
  Description: Constructor for the simple IGM state class.
  Code lines: 13
  Contained by: file intergalactic_medium.state.simple.F90
  Modules used: cosmology_functions

function: simpledefaultconstructor
  Description: Default constructor for the simple IGM state class.
  Code lines: 50
  Contained by: file intergalactic_medium.state.simple.F90
  Modules used: input_parameters

function: simpleelectronfraction
  Description: Return the electron fraction of the IGM in the simple model.
  Code lines: 13
  Contained by: file intergalactic_medium.state.simple.F90
  Modules used: numerical_constants_astronomical

function: simpleneutralheliumfraction
  Description: Return the neutral helium fraction of the IGM in the simple model.
  Code lines: 13
  Contained by: file intergalactic_medium.state.simple.F90
  Modules used: numerical_constants_astronomical

function: simpleneutralhydrogenfraction
  Description: Return the neutral hydrogen fraction of the IGM in the simple model.
  Code lines: 13
  Contained by: file intergalactic_medium.state.simple.F90
  Modules used: numerical_constants_astronomical

function: simplesinglyionizedheliumfraction
  Description: Return the singly-ionized helium fraction of the IGM in the simple model.
  Code lines: 13
  Contained by: file intergalactic_medium.state.simple.F90
  Modules used: numerical_constants_astronomical

function: simpletemperature
  Description: Return the temperature of the IGM in the simple model.
  Code lines: 12
  Contained by: file intergalactic_medium.state.simple.F90

file: iso_varying_string.F90
module: iso_varying_string

Code lines: 2674

Code lines: 2633

Contained by: file iso_varying_string.F90

Used by:

  program conditional_mass_function
  program galacticus
  program mass_function_covariance
  program simple_merge_tree_file_maker
  function accretionhaloconstructor
  module accretion_disks_adaf

module accretion_disks

module accretion_disk_spectra

subroutine accretion disks spectra constructor

module atomic_cross_sections_ionization_photo

subroutine atomic_data_initialize

subroutine atomic_ionization_potentials_vernernize_initialize

subroutine gaunt_factor_sutherland1998_initialize

subroutine collisional_excitation_cooling_rate_scholzwalters91_initialize

subroutine atomic_rate_ionization_collisional_vernernize_initialize

module atomic_rates_recombination_dielectronic

subroutine atomic_rate_recombination_radiative_vernernize_initialize

subroutine black_hole_binary_initial_radii_volonteri_2003_initialize

subroutine black_hole_binary_initial_radii_tidal_radius_initialize

module black_hole_binary_recoil_velocities

module black_hole_binary_separations

program tests_excursion_sets

program halo_mass_functions

program power_spectra

module accretion_halos

subroutine accretionhaloinitialize

subroutine accretion_disks_eddington_initialize

subroutine accretion_disks_shakura_sunyaev_initialize

function accretion_disks_spectra_constructor

subroutine accretion_disks_switched_initialize

subroutine atomic_cross_section_ionization_photo_vernernize_initialize

module atomic_rates_recombination_radiative

module black_hole_binary_recoil_velocity_standard_initialize

module black_hole_binary_recoil_velocity_null_initialize

subroutine black_hole_binary_recoil_velocity_standard

subroutine black_hole_binary_recoil_velocity_null

subroutine black_hole_binary_separation_growth_rate_null_initialize
19.1. Program units

subroutine black_hole_binary_separation_growth_rate_standard_init
subroutine black_hole_binary_merger_rezzolla_initialize
subroutine chemical_hydrogen_rates_initialize
module chemical_states_cie_file
module chemical_states_atomic_cie_cloudy
subroutine cooling_cold_mode_infall_rate_dynamical_time_initialize
module cooling_functions_cie_file
subroutine cooling_functions_null_initialize
module cooling_functions_atomic_cie_cloudy
subroutine cooling_functions_atom_cloudy
module cooling_radii
subroutine cooling_radius_simple_initialize
subroutine cooling_rate_cole2000_initialize
subroutine cooling_rate_white_frenk_initialize
subroutine cooling_rate_simple_initialize
subroutine cooling_rate_zero_initialize
subroutine cooling_time_simple_initialize
subroutine cooling_rate_modifier_cut_off
subroutine cooling_rate_simple_scaling_initialize
subroutine cooling_rate_zero_initialize
module cooling_freefall_times_available
subroutine cooling_time_simple_initialize
subroutine cooling_time_simple_scaling_initialize
module cooling_times
subroutine cooling_freefall_time_available_haloidinitialize
subroutine infall_radius_cooling_freefall_initialize
module cooling_specific_angular_momenta
subroutine cooling_specific_am_mean_initialize
subroutine cooling_time_available_wf_haloidinitialize
module cosmology_functions
subroutine cosmologyfunctionsinitialize
function matterdarkenergyconstructor
function matterlambdaconstructor
module cosmology_parameters
subroutine cosmologyparametersinitialize
module dark_matter_halo_mass_accretion_histories
subroutine dark_matter_mass_accretion_wechsler2002_initialize
module dark_matter_halos_mass_loss_rates
subroutine dark_matter_halos_mass_loss_rate_vandenbosch_initialize
function darkmatterhaloscaleconstructornamed
subroutine dark_matter_halo_scales

subroutine darkmatterhaloscaleinitialize
subroutine halo_spin_distribution_bett2007_initialize
subroutine halo_spin_distribution_delta_function_initialize
module dark_matter_profiles

subroutine darkmatterprofileinitialize
function darkmatterprofileconcentrationconstructornamed
module dark_matter_profiles_shapes

function node_branch_jump
subroutine events_node_merger_do_slh
subroutine galactic_dynamics_bar_instabilities_eln_initialize
module galactic_dynamics_bar_instabilities
function galactic_structure_radius_enclosing_mass
subroutine galactic_structure_radii_adiabatic_initialize
subroutine galactic_structure_radii_fixed_initialize
subroutine galactic_structure_initial_radii_adiabatic_initialize
subroutine galactic_structure_radii_linear_initialize
module galacticus_display
subroutine galacticus_error_report_varstr
module galacticus_meta_compute_times
subroutine galacticus_time_per_tree_file_initialize
19.1. Program units

subroutine galacticus_meta_evolver_profile
module galacticus_output_open
subroutine galacticus_output_analysis_mass_dpndnt_sz_dstrbtins
subroutine load_primus_mass_function
function galacticus_build_string
subroutine galacticus_output_tree_descendents_initialize
subroutine galacticus_merge_tree_output_filter_lightcone_initialize
subroutine galacticus_output_tree_final_descendents_initialize
subroutine galacticus_output_tree_half_mass_names
module galacticus_output_trees_rotation_curve
module galacticus_output_star_formation_histories
subroutine star_formation_history_output_metallicity_split
module galacticus_output_trees_velocity_dispersion
subroutine galacticus_version_output
function galacticus_task_evolve_tree

subroutine geometry_surveys_caputi_2011_ukidss_uds
subroutine geometry_survey_window_functions_caputi_2011_ukidss_uds_initialize
subroutine geometry_survey_window_functions_li_white_2009_sdss
subroutine geometry_survey_window_functions_martin_2010_alfalfa
subroutine halo_mass_function_compute
subroutine conditional_mass_functions_behroozi2010_initialize
module hot_halo_cold_mode_density_cored_isothermal_core_radii
module hot_halo_mass_distributions

subroutine hothalomassdistributioninitialize
function hothalomassdistributioncoreradiusinitialize
module hot_halo_mass_distributions_core_radii
<table>
<thead>
<tr>
<th>Module/Function</th>
<th>Subroutine</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>hot_halo_ram_pressure_forces</code></td>
<td><code>hot_halo_ram_pressure_force2008_initialize</code></td>
</tr>
<tr>
<td><code>hot_halo_ram_pressure_force_null_initialize</code></td>
<td></td>
</tr>
<tr>
<td><code>hot_halo_ram_pressure_stripping_font2008_initialize</code></td>
<td></td>
</tr>
<tr>
<td><code>hot_halo_ram_pressure_stripping_timescales_halo_dyntime_initialize</code></td>
<td></td>
</tr>
<tr>
<td><code>hot_halo_ram_pressure_stripping_virial_radii_initialize</code></td>
<td></td>
</tr>
<tr>
<td><code>hot_halo_temperature_virial</code></td>
<td></td>
</tr>
<tr>
<td><code>intergalactic_medium_state</code></td>
<td><code>intergalacticmediumstateinitialize</code></td>
</tr>
<tr>
<td><code>inverse_gamma_function_incomplete_complementary</code></td>
<td></td>
</tr>
<tr>
<td><code>generalized_press_schechter_branch_mass</code></td>
<td><code>generalized_press_schechter_branching_initialize</code></td>
</tr>
<tr>
<td><code>generalized_press_schechter_subresolution_fraction</code></td>
<td><code>generalized_press_schechter_subresolution_fraction</code></td>
</tr>
<tr>
<td><code>merger_tree_branching_modifiers</code></td>
<td><code>merger_tree_branching_modifiers_parkinson_initialize</code></td>
</tr>
<tr>
<td><code>merger_tree_branching_modifiers_null_initialize</code></td>
<td><code>merger_tree_branching_modifiers_parkinson_initialize</code></td>
</tr>
<tr>
<td><code>merger_tree_branching_modifiers_null_initialize</code></td>
<td></td>
</tr>
<tr>
<td><code>merger_tree_construction</code></td>
<td></td>
</tr>
<tr>
<td><code>merger_trees_build_mass_resolution_fixed_initialize</code></td>
<td><code>merger_trees_build_mass_resolution_fixed_initialize</code></td>
</tr>
<tr>
<td><code>merger_trees_build_mass_resolution_scaled_initialize</code></td>
<td><code>merger_trees_build_mass_resolution_scaled_initialize</code></td>
</tr>
<tr>
<td><code>merger_trees_constructFullySpecified</code></td>
<td></td>
</tr>
<tr>
<td><code>merger_trees_mass_function_sampling_gaussian_initialize</code></td>
<td><code>merger_trees_mass_function_sampling_gaussian_initialize</code></td>
</tr>
<tr>
<td><code>merger_trees_mass_function_sampling_power_law_initialize</code></td>
<td><code>merger_trees_mass_function_sampling_power_law_initialize</code></td>
</tr>
<tr>
<td><code>merger_tree_read</code></td>
<td><code>merger_tree_read_importers</code></td>
</tr>
<tr>
<td><code>mergetreeimporterinitialize</code></td>
<td></td>
</tr>
<tr>
<td><code>merger_tree_smooth_accretion</code></td>
<td></td>
</tr>
<tr>
<td><code>merger_trees_dump_evolution</code></td>
<td></td>
</tr>
<tr>
<td><code>merger_trees_dump_structure</code></td>
<td></td>
</tr>
<tr>
<td><code>merger_trees_state_store</code></td>
<td></td>
</tr>
<tr>
<td><code>merger_trees_construction</code></td>
<td></td>
</tr>
<tr>
<td><code>merger_trees_build_mass_resolution_fixed_initialize</code></td>
<td><code>merger_trees_build_mass_resolution_fixed_initialize</code></td>
</tr>
<tr>
<td><code>merger_trees_build_mass_resolution_scaled_initialize</code></td>
<td><code>merger_trees_build_mass_resolution_scaled_initialize</code></td>
</tr>
<tr>
<td><code>merger_trees_constructFullySpecified</code></td>
<td><code>merger_trees_constructFullySpecified</code></td>
</tr>
<tr>
<td><code>merger_trees_mass_function_sampling_gaussian_initialize</code></td>
<td><code>merger_trees_mass_function_sampling_gaussian_initialize</code></td>
</tr>
<tr>
<td><code>merger_trees_mass_function_sampling_power_law_initialize</code></td>
<td><code>merger_trees_mass_function_sampling_power_law_initialize</code></td>
</tr>
<tr>
<td><code>merger_tree_read</code></td>
<td><code>merger_tree_read_importers</code></td>
</tr>
<tr>
<td><code>mergetreeimporterinitialize</code></td>
<td></td>
</tr>
<tr>
<td><code>merger_tree_smooth_accretion</code></td>
<td></td>
</tr>
<tr>
<td><code>merger_trees_dump_evolution</code></td>
<td></td>
</tr>
<tr>
<td><code>merger_trees_dump_structure</code></td>
<td></td>
</tr>
</tbody>
</table>
module merger_trees_evolve
function time_step_get
subroutine merger_tree_record_evolution_output
subroutine evolve_to_time_report
subroutine satellite_merger_process
subroutine merger_trees_bolshoi_process
subroutine merger_trees_simple_process
module merger_trees_structure_output
module merger_trees_write
function interpolate
function interpolate_derivative
function root_finder_find
function search_array_varstring
module abundances_structure
module chemical_structures
subroutine history_extend
subroutine kepler_orbits_dump
subroutine merger_tree_dump
module galacticus_nodes

module merger_trees_evolve_node
subroutine merger_tree_timestep_history
subroutine merger_tree_timestep_record_evolution
subroutine merger_tree_timestep_satellite
subroutine merger_tree_timestep_simple
subroutine merger_tree_mass_accretion_history_output
subroutine odeiv2_solve
function interpolate_derivative
function search_array_varstring
module chemical_abundances_structure
subroutine history_dump
subroutine history_long_integer_dump
module merger_tree_data_structure

function agestatisticsdiskintegratedsfrattributematch
function agestatisticsdisktimeweightedintegratedsfrattributematch
function agestatisticsdiskintegratingardintegrationattributematch
function agestatisticsspheroidintegratedsfrattributematch
function agestatisticsspheroidtimeweightedintegratedsfrattributematch
function basicaccretionrateattributematch
function basicmassattributematch
function basicmassmaximumattributematch
function basictimelastisolatedattributematch
function blackholeaccretionrateattributematch
function blackholemassattributematch
function blackholemassseedattributematch
function blackholeradialpositionattributematch
function blackholeradiativeefficiencyattributematch
function blackholespinattributematch
function blackholespinseedattributematch
function blackholetripleinteractiontimeattributematch
function darkmatterprofilescaleattributematch
function darkmatterprofileshapeattributematch
function darkmatterprofilescalegrowthrateattributematch
function darkmatterprofileshapegrowthrateattributematch
function diskabundancesgasattributematch
function diskabundancesstellarattributematch
function diskangularmomentumattributematch
19. Source Code Documentation

subroutine diskcreatelinked
function diskhalfmassradiusattributematch
function diskisinitializedattributematch
diskisinitializedattributematch
diskmassgasattributematch
diskmassgasattributematch
diskmassstellarattributematch
diskmassstellarattributematch
diskradiusattributematch
diskradiusattributematch
diskstarformationhistoryattributematch
diskstarformationhistoryattributematch
diskstarformationrateattributematch
diskstarformationrateattributematch
diskstellarpropertieshistoryattributematch
diskstellarpropertieshistoryattributematch
diskvelocityattributematch
diskvelocityattributematch
dynamicsstatisticsadiabaticratioattributematch
dynamicsstatisticsadiabaticratioattributematch
dynamicsstatisticsbarinstabilitytimescaleattributematch
dynamicsstatisticsbarinstabilitytimescaleattributematch
subroutine dynamicsstatisticscreatelinked
function dynamicsstatisticstimeattributematch
subroutine formationtimecreatelinked
function formationtimeformationtimeattributematch
subroutine hosthistorycreatelinked
function hosthistoryhostmassmaximumattributematch
function hosthistoryhostmassmaximumattributematch
function hostmassmaximumattributematch
function hostmassmaximumattributematch
function hostmasscoldattributematch
hostmasscoldattributematch
function hostmasssinkattributematch
subroutine hothalocreatelinked
function hothaloheatsourceattributematch
function hothaloheatsourceattributematch
function hothalochemicalsattributematch
function hothalochemicalsattributematch
function hothaloisinitializedattributematch
function hothaloisinitializedattributematch
function hothalomassattributematch
function hothalomassattributematch
function hothalomasscoldattributematch
function hothalomasscoldattributematch
function hothaloouterradiustimeattributematch
function hothaloouterradiustimeattributematch
function hothalooutflowedmassattributematch
function hothalooutflowedmassattributematch
function hothalooutflowedangularmomentumattributematch
function hothalooutflowedangularmomentumattributematch
function hothalooutflowingmassattributematch
function hothalooutflowingmassattributematch
function hothalooutflowingabundancesattributematch
function hothalooutflowingabundancesattributematch
function hothalostrippendmassattributematch
function hothalostrippendmassattributematch
function hothaloutrackedoutflowmassattributematch
function hothaloutrackedoutflowmassattributematch
function hothalounaccretedmassattributematch
function hothalounaccretedmassattributematch
function indicesbranchtipattributematch
subroutine indicescreatelinked
function interoutputdiskstarformationrateattributematch
function interoutputdiskstarformationrateattributematch
19.1. Program units

function interoutputspheroidstarformationrateattributematch
subroutine massflowstatisticscreatelinked
function massflowstatisticscooledmassattributematch
function mergingstatisticscreatelinked
function mergingstatisticsgalaxymajormajormergerstatisticstimeattributematch
function mergingstatisticsnodeformationtimeattributematch
function mergingstatisticsnodehierarchylevelattributematch
function mergingstatisticsnodehistoryrecentmajormergercountattributematch
subroutine nbodycreatelinked
subroutine nbodyparticlecountattributematch
function nbodyvelocitydispersionattributematch
function nbodyvelocitymaximumattributematch
subroutine node_component_blackhole_dump
subroutine node_component_blackhole_null_dump
subroutine node_component_blackhole_simple_dump
subroutine node_component_blackhole_standard_dump
subroutine node_component_blackholenull_dump
subroutine node_component_blackholenull_name_from_index
subroutine node_component_blackholesimple_dump
subroutine node_component_blackholesimple_name_from_index
subroutine node_component_blackholestandard_dump
subroutine node_component_blackholestandardtracking_dump
subroutine node_component_blackhole_standard_dump
subroutine node_component_blackhole_standard_name_from_index
subroutine node_component_blackhole_standard_tracking_dump
subroutine node_component_blackholestandard_name_from_index
subroutine node_component_blackhole_standardtracking_name_from_index
subroutine node_component_blackholenull_name_from_index
subroutine node_component_blackholenull_name_from_index
subroutine node_component_blackhole_standard_name_from_index
subroutine node_component_blackhole_standard_name_from_index
subroutine node_component_blackhole_standard_name_from_index
subroutine node_component_blackhole_standard_name_from_index
subroutine node_component_blackhole_standard_name_from_index
subroutine node_component_blackhole_standard_name_from_index
subroutine node_component_darkmatterprofilescaleshape_name_from_index subroutine node_component_disk_dump
subroutine node_component_diskexponential_dump subroutine node_component_disknull_dump subroutine node_component_diskverysimple_dump subroutine node_component_dynamicsstatistics_dump subroutine node_component_dynamicsstatisticsbars_dump subroutine node_component_dynamicsstatisticsnull_name_from_index subroutine node_component_formationtime_dump subroutine node_component_formationtimecole2000_dump subroutine node_component_formationtimenull_dump subroutine node_component_hosthistory_formationtimecole2000_name_from_index subroutine node_component_hosthistorynull_name_from_index subroutine node_component_hosthistorystandard_name_from_index subroutine node_component_hothalo_hothalocoldmode_dump subroutine node_component_hothalonull_name_from_index subroutine node_component_hothalocoldmode_name_from_index subroutine node_component_hothaloverysimple_dump subroutine node_component_hothaloverysimple_name_from_index subroutine node_component_indices_dump subroutine node_component_indicesnull_name_from_index subroutine node_component_indicesnull_name_from_index subroutine node_component_interoutput_indicesstandard_name_from_index subroutine node_component_interoutputnull_name_from_index subroutine node_component_interoutputnull_name_from_index subroutine node_component_hosthistorystandard_name_from_index
subroutine node_component_interoutputnull_dump
genode_interoutputnull_name_from_index
subroutine node_component_interoutputstandard_dump
genode_interoutputstandard_name_from_index
subroutine node_component_massflowstatistics_dump
ngenode_massflowstatisticsnull_name_from_index
subroutine node_component_massflowstatisticsstandard_dump
genode_massflowstatisticsstandard_name_from_index
subroutine node_component_mergingstatistics_dump
ngenode_mergingstatisticsnull_name_from_index
subroutine node_component_mergingstatisticsrecent_dump
genode_mergingstatisticsrecent_name_from_index
subroutine node_component_mergingstatisticsstandard_dump
genode_mergingstatisticsstandard_name_from_index
subroutine node_component_name_from_index
subroutine node_component_nbody_dump
ngenode_nbodynull_name_from_index
subroutine node_component_nbodystandard_dump
genode_nbodystandard_name_from_index
subroutine node_component_position_dump
ngenode_positionnull_name_from_index
subroutine node_component_positionpreset_name_from_index
subroutine node_component_satellitenull_dump
ngenode_satellitenull_name_from_index
subroutine node_component_satelliteorbiting_dump
ngenode_satelliteorbiting_name_from_index
subroutine node_component_satellitepreset_dump
ngenode_satellitepreset_name_from_index
subroutine node_component_satellitestandard_dump
ngenode_satellitestandard_name_from_index
subroutine node_component_satelliteverysimple_dump
ngenode_satelliteverysimple_name_from_index
subroutine node_component_spheroid_dump
ngenode_spheroidnull_dump
subroutine node_component_spheroidnull_name_from_index
subroutine node_component_spheroidstandard_dump
subroutine node_component_spheroidstandard_name_from_index
subroutine node_component_spheroidnull_dump
subroutine node_component_spin_dump
subroutine node_component_spinnull_name_from_index
subroutine node_component_spinnull_dump
subroutine node_component_spinpreset3d_dump
subroutine node_component_spinpreset_name_from_index
subroutine node_component_spinpreset_dump
subroutine node_component_spinpreset_name_from_index
subroutine node_component_spinnrandom_dump
subroutine node_component_spinnrandom_name_from_index
subroutine node_dump
subroutine positioncreatelinked
function node_property_name_from_index
function positionpositionattributematch
function positionvelocityattributematch
function satelliteboundmasshistoryattributematch
function satellitemergetimeattributematch
function satellitenodeindexattributematch
function satellitetidalheatingnormalizedattributematch
function satellitetimeofmergingattributematch
function satellitevirialorbitattributematch
function spheroidabundancesgasattributematch
subroutine spheroidcreatelinked
function satellitenodeindexhistoryattributematch
function satellitepositionattributematch
function satellitetidaltensorpathintegratedattributematch
function spheroidabundancesstellarattributematch
function spheroidenergygasinputattributematch
function spheroidhalfmassradiusattributematch
function spheroidluminositiesstellarattributematch
subroutine spheroidcreatelinked
function spheroidangularmomentumattributematch
function spheroidenergygasinputattributematch
function spheroidmasssinkattributematch
function spheroidmassstellarattributematch
function spheroidradiusattributematch
function spheroidstarformationhistoryattributematch
function spheroidstarformationrateattributematch
function spheroidstellarpropertieshistoryattributematch
19.1. Program units

function spheroidvelocityattributematch
function spinnspinattributematch

function spinnspinvectormatch

subroutine node_component_black_hole_-standard_output_properties
module node_component_disk_very_simple

subroutine node_component_disk_-statistics_bars_timestep
subroutine node_component_hot_halo_-standard_initialize
subroutine node_component_spheroid_-standard_initialize
module stellar_luminosities_structure
subroutine tensor_r2_d3_sym_dump
subroutine tensor_r2_d3_sym_dump_raw
subroutine tensor_r2_d3_sym_read_raw

subroutine radiation_igb_file_-initialize
function ram_pressure_stripping_mass_-loss_rate_disk
subroutine ram_pressure_stripping_-mass_loss_rate_disks_simple_init
subroutine ram_pressure_stripping_-mass_loss_rate_spheroids_null_init
subroutine satellite_dynamical_-friction_chandrasekhar_initialize
subroutine satellite_dynamical_-friction_acceleration_null_initialize
module satellite_merging_mass_-movements
subroutine satellite_merging_mass_-movements_simple_initialize
subroutine satellite_merging_remnant_-sizes_cole2000 Initialize
subroutine satellite_merging_remnant_-sizes_covington2008_initialize
subroutine satellite_merging_remnant_-sizes_null_initialize
module satellite_merging_remnant_-sizes_progenitors
subroutine spincreatelinked
function spinnspinvgrowthrateattributematch

function spinnspinvectorgrowthrateattributematch
subroutine node_component_disk_-exponential_post_evolve
subroutine node_component_dynamics_-statistics_bars_output
module node_component_hot_halo_cold_-mode
subroutine node_component_merging_-statistics_recent_initialize
subroutine node_component_spheroid_-standard_post_evolve
module tensors
subroutine tensor_r2_d3_sym_dump_raw
program optimal_sampling_smf
subroutine radiation_initialize_-intergalactic_background
function radiation_intergalactic_-background_internal_update
subroutine ram_pressure_stripping_-mass_loss_rate_disks_null_init
function ram_pressure_stripping_-mass_loss_rate_spheroid
subroutine ram_pressure_stripping_-mass_loss_rate_spheroids_simple_init
module satellite_dynamical_friction
subroutine satellite_merging_mass_-movements_baugh2005_initialize
subroutine satellite_merging_mass_-movements_simple_initialize
subroutine satellite_merging_remnant_-size_cole2000
subroutine satellite_merging_remnant_-size_covington2008
module satellite_merging_remnant_sizes
subroutine satellite_merging_remnant_-progenitor_properties_cole2000_init
subroutine satellite_merging_remnant_-progenitor_properties_standard_init
module satellite_merging_timescales
subroutine satellite_merging_timescalesinitialize
module virial_orbits
subroutine virial_orbital_parameters_-fixed_initialize
module satellites_tidal_fields
subroutine satellites_tidal_fields_-spherical_symmetry_initialize
subroutine satellite_tidal_heating_-rate_gnedin_initialize
module satellite_tidal_stripping
subroutine satellite_tidal_stripping_-rate_null_initialize
subroutine star_formation_imf_-register_name_chabrier
subroutine star_formation_imf_-register_name_kennicutt
subroutine star_formation_imf_-register_name_millerscalo
subroutine satellitemergingtimescalesconstructornamed
subroutine star_formation_imf_-register_name_salpeter
subroutine star_formation_imf_-register_name_piecewisepowerlaw
subroutine imf_select_disk_spheroid_-initialize
function imf_index_lookup
module star_formation_feedback_disks
subroutine star_formation_feedback_-disks_halo_scaling_initialize
module star_formation_feedback_-spheroids
subroutine star_formation_feedback_-expulsion_disks
subroutine star_formation_expulsive_-feedback_disks_sw_initialize
subroutine star_formation_expulsive_-feedback_spheroids_null_initialize
subroutine star_formation_rate_-surface_density_disks_br_initialize
subroutine star_formation_feedback_-disks_creasey2012_initialize
subroutine star_formation_feedback_-disks_fixed_initialize
subroutine star_formation_feedback_-disks_power_law_initialize
subroutine star_formation_feedback_-spheroids_power_law_initialize
subroutine star_formation_expulsive_-feedback_disks_null_initialize
subroutine star_formation_expulsive_-expulsion_spheroids
subroutine star_formation_rate_-surface_density_disks
subroutine star_formation_rate_-surface_density_disks_kmt09_-initialize
subroutine star_formation_rate_-surface_density_disks_ks_initialize
subroutine star_formation_rate_-surface_density_disks_exschmidt_-initialize
subroutine star_formation_timescale_-disks_dynamical_time_initialize
subroutine star_formation_timescale_-disks_halo_scaling_initialize
module star_formation_timescales_-spheroids
module stellar_astrophysics
subroutine stellar_feedback_standard_-initialize
module supernovae_population_iii
module supernovae_type_ia
module stellar_astrophysics_tracks
module stellar_astrophysics_winds
function stellar_population_luminosity
module stellar_population_properties
module stellar_population_spectra_-conroy
module stellar_population_spectra_file
function spectrapostprocessorconstructornamed
subroutine cosmological_mass_variance_-filtered_power_spectrum_initialize
subroutine critical_overdensity_-kitayama_suto1996_initialize
subroutine critical_overdensity_mass_-scaling_wdm_initialize
subroutine excursion_sets_barriers_-critical_overdensity_initialize
subroutine excursion_sets_barriers_-critical_overdensity_mass_-scaling_wdm_initialize
subroutine excursion_sets_barriers_-critical_overdensity_mass_-scaling_null_initialize
subroutine excursion_sets_barriers_-linear_initialize
subroutine excursion_sets_barriers_-quadratic_initialize
subroutine excursion_sets_barriers_-remap_smt_initialize
subroutine excursion_sets_barriers_-remap_null_initialize
subroutine excursion_sets_barriers_-remap_scale_initialize
module star_formation_timescales_disks
module star_formation_timescales_spheroids
subroutine star_formation_timescale_-disks_fixed_initialize
subroutine star_formation_timescale_-disks_integrated_sd_initialize
subroutine star_formation_timescale_-spheroids_dynamical_time_initialize
module stellar_feedback
subroutine stellarAstrophysics_file_-initialize
subroutine supernovae_population_iii_-hegerwoosley_initialize
subroutine supernovae_type_ia_-nagashima_initialize
subroutine stellar_tracks_initialize_-file
subroutine stellar_winds_-leitherer1992_initialize
module stellar_population_properties
subroutine stellar_population_-properties_noninstantaneous_-initialize
module stellar_population_spectra
module stellar_population_spectra_-postprocess
subroutine
spectrapostprocessorinitialize
module critical_overdensity
subroutine critical_overdensity_mass_-scaling_null_initialize
module excursion_sets_barriers
subroutine excursion_sets_barriers_-linear_initialize
subroutine excursion_sets_barriers_-remap_smt_initialize
subroutine excursion_sets_barriers_-remap_null_initialize
subroutine excursion_sets_barriers_-remap_scale_initialize
module excursion_sets_first_crossings
subroutine excursion_sets_first_crossing_zhang_hui_initialize
subroutine excursion_sets_first_crossing_linear_barrier_initialize
subroutine dark_matter_halo_bias_press_schechter_initialize
subroutine dark_matter_halo_bias_tinker2010_initialize
subroutine halo_mass_function_press_schechter_initialize
subroutine halo_mass_function_tinker2008_initialize
subroutine growth_factor_simple_initialize
function power_spectrum_nonlinear_cosmicemu
module power_spectra_nonlinear
subroutine power_spectrum_nonlinear_peacockdodds1996_initialize
subroutine power_spectrum_nonlinear_linear_initialize
subroutine primordial_power_spectrum_power_law_initialize
subroutine power_spectrum_window_functions
subroutine power_spectrum_window_functions_sharp_kspace_initialize
subroutine power_spectrum_window_functions_th_kss_hybrid_initialize
subroutine sphericalCollapse_delta_critical_initialize
module transfer_function_camb
module transfer_functions
subroutine transfer_function_null_initialize
function virialdensitycontrastconstructormamed
subroutine system_command_do
program test_nfw96_concentration_dark_energy
program test_zhao2009_flat
program test_zhao2009_open
program test_array_monotonicity
program tests_comoving_distance
program tests_halo_mass_function_tinker
module excursion_sets_first_crossing_farahi
subroutine excursion_sets_first_crossing_zhang_hui_high_initialize
module dark_matter_halo_biases
subroutine dark_matter_halo_bias_smt_initialize
module halo_mass_function
subroutine halo_mass_function_sheth_tormen_initialize
module linear_growth
subroutine power_spectra
module power_spectra_nonlinear
subroutine power_spectrum_nonlinear_cosmicemu_initialize
subroutine power_spectrum_nonlinear_peacockdodds1996_initialize
module primordial_power_spectra
module power_spectrum_window_functions
subroutine power_spectrum_window_functions_top_hat_initialize
module sphericalCollapse_matter_dark_energy
module transfer_function_bbks
module transfer_function_eisenstein_hu
module transfer_functions_file
module virial_density_contrast
subroutine virialdensitycontrastinitialize
program tests_io_hdf5
program test_prada2011_concentration
program test_zhao2009_dark_energy
program test_abundances
program tests_bug745815
program tests_cosmic_age
program test_hashes_cryptographic
19.1. Program units

program tests_kepler_orbits
program tests_linear_growth_cosmological_constant
program tests_linear_growth_open
subroutine testvoidfunc
program test_search
program tests_spherical_collapse_dark_energy_eds
program tests_spherical_collapse_dark_energy_omega_zero_point_eight
program tests_spherical_collapse_dark_energy_omega_two_thirds
program tests_spherical_collapse_dark_energy_open
program tests_spherical_collapse_open
function tidal_stripping_mass_loss_rate_disk
subroutine tidal_stripping_mass_loss_rate_disks_null_init
subroutine tidal_stripping_mass_loss_rate_disks_simple_init
subroutine tidal_stripping_mass_loss_rate_spheroids_null_init
module io_hdf5

subroutine get_argument_varying_string
function formatted_date_and_time
module hashes
module input_parameters
subroutine alloc_array_character_1d
subroutine alloc_array_double_precision_1d
subroutine alloc_array_double_precision_2d
subroutine alloc_array_double_precision_3d
subroutine alloc_array_double_precision_4d
subroutine alloc_array_double_precision_5d
subroutine alloc_array_integer_1d
subroutine alloc_array_integer_2d
subroutine alloc_array_integer_kind_int8_1d

program tests_linear_growth_eds
program tests_linear_growth_dark_energy
program test_nodes
program tests_power_spectrum
program tests_sigma
program tests_spherical_collapse_dark_energy_omega_zero_point_five
program tests_spherical_collapse_dark_energy_omega_half
program tests_spherical_collapse_dark_energy_omega_two_thirds
program tests_spherical_collapse_dark_energy_omega_zero_point_eight
program tests_spherical_collapse_dark_energy_open
program tests_spherical_collapse_flat

program test_string_utilities
subroutine tidal_stripping_mass_loss_rate_disks_null_init
function tidal_stripping_mass_loss_rate_spheroid
subroutine tidal_stripping_mass_loss_rate_spheroids_null_init
function array_intersection_varying_string
subroutine get_temporary_string
module file_utilities
module hashes
subroutine add_memory_component
subroutine alloc_array_character_1d_kind_int8
subroutine alloc_array_double_precision_1d_kind_int8
subroutine alloc_array_double_precision_2d_kind_int8
subroutine alloc_array_double_precision_3d_kind_int8
subroutine alloc_array_double_precision_4d_kind_int8
subroutine alloc_array_double_precision_5d_kind_int8
subroutine alloc_array_integer_1d_kind_int8
subroutine alloc_array_integer_2d_kind_int8
subroutine alloc_array_integer_kind_int8_1d_kind_int8
subroutine alloc_array_integer_kind_int8_2d
subroutine alloc_array_logical_1d
subroutine alloc_array_real_1d
subroutine code_memory_usage
subroutine dealloc_array_double_precision_1d
subroutine dealloc_array_double_precision_3d
subroutine dealloc_array_double_precision_5d
subroutine dealloc_array_integer_2d
subroutine dealloc_array_integer_kind_int8_2d
subroutine dealloc_array_real_1d
subroutine dealloc_array_character_1d
subroutine dealloc_array_double_precision_2d
subroutine dealloc_array_double_precision_4d
subroutine dealloc_array_double_precision_5d
subroutine dealloc_array_integer_kind_int8_1d
subroutine dealloc_array_integer_kind_int8_2d
subroutine dealloc_array_logical_1d
subroutine dealloc_array_real_1d
subroutine memory_usage_record
module unit_tests

interface: adjustl
Code lines: 2
Contained by: module iso_varying_string

function: adjustl_
Code lines: 13
Contained by: module iso_varying_string

interface: adjustr
Code lines: 2
Contained by: module iso_varying_string

function: adjustr_
Code lines: 13
Contained by: module iso_varying_string

interface: assignment(=)
Code lines: 3
Contained by: module iso_varying_string

interface: char
Code lines: 3
Contained by: module iso_varying_string

function: char_auto
Code lines: 18
Contained by: module iso_varying_string
19.1. Program units

**function:** char_fixed

*Code lines:* 15  
*Contained by:* module iso_varying_string

**subroutine:** destroy_vs

*Description:* Destroy a varying string object by deallocating it. Can be necessary to avoid memory leaks in some instances.  
*Code lines:* 6  
*Contained by:* module iso_varying_string

**interface:** extract

*Code lines:* 3  
*Contained by:* module iso_varying_string

**function:** extract_ch

*Code lines:* 30  
*Contained by:* module iso_varying_string

**function:** extract_vs

*Code lines:* 15  
*Contained by:* module iso_varying_string

**interface:** get

*Code lines:* 7  
*Contained by:* module iso_varying_string

**subroutine:** get_

*Code lines:* 51  
*Contained by:* module iso_varying_string

**subroutine:** get_set_ch

*Code lines:* 58  
*Contained by:* module iso_varying_string

**subroutine:** get_set_vs

*Code lines:* 22  
*Contained by:* module iso_varying_string

**subroutine:** get_unit

*Code lines:* 53  
*Contained by:* module iso_varying_string

**subroutine:** get_unit_set_ch

*Code lines:* 60  
*Contained by:* module iso_varying_string

**subroutine:** get_unit_set_vs

*Code lines:* 23  
*Contained by:* module iso_varying_string
interface: iachar
Code lines: 2
Contained by: module iso_varying_string

function: iachar_
Code lines: 14
Contained by: module iso_varying_string

interface: ichar
Code lines: 2
Contained by: module iso_varying_string

function: ichar_
Code lines: 14
Contained by: module iso_varying_string

interface: index
Code lines: 4
Contained by: module iso_varying_string

function: index_ch_vs
Code lines: 16
Contained by: module iso_varying_string

function: index_vs_ch
Code lines: 16
Contained by: module iso_varying_string

function: index_vs_vs
Code lines: 16
Contained by: module iso_varying_string

interface: insert
Code lines: 5
Contained by: module iso_varying_string

function: insert_ch_ch
Code lines: 20
Contained by: module iso_varying_string

function: insert_ch_vs
Code lines: 15
Contained by: module iso_varying_string

function: insert_vs_ch
Code lines: 15
Contained by: module iso_varying_string

function: insert_vs_vs
19.1. Program units

Code lines: 15
Contained by: module iso_varying_string

interface: len
Code lines: 2
Contained by: module iso_varying_string

function: len_
Code lines: 17
Contained by: module iso_varying_string

interface: len_trim
Code lines: 2
Contained by: module iso_varying_string

function: len_trim_
Code lines: 17
Contained by: module iso_varying_string

interface: lge
Code lines: 4
Contained by: module iso_varying_string

function: lge_ch_vs
Code lines: 15
Contained by: module iso_varying_string

function: lge_vs_ch
Code lines: 15
Contained by: module iso_varying_string

function: lge_vs_vs
Code lines: 14
Contained by: module iso_varying_string

interface: lgt
Code lines: 4
Contained by: module iso_varying_string

function: lgt_ch_vs
Code lines: 15
Contained by: module iso_varying_string

function: lgt_vs_ch
Code lines: 15
Contained by: module iso_varying_string

function: lgt_vs_vs
Code lines: 14
19. Source Code Documentation

**Contained by:** module `iso_varying_string`

**interface: l1le**
- **Code lines:** 4
- **Contained by:** module `iso_varying_string`

**function: l1le_ch_vs**
- **Code lines:** 15
- **Contained by:** module `iso_varying_string`

**function: l1le_vs_ch**
- **Code lines:** 15
- **Contained by:** module `iso_varying_string`

**function: l1le_vs_vs**
- **Code lines:** 14
- **Contained by:** module `iso_varying_string`

**interface: l1lt**
- **Code lines:** 4
- **Contained by:** module `iso_varying_string`

**function: l1lt_ch_vs**
- **Code lines:** 15
- **Contained by:** module `iso_varying_string`

**function: l1lt_vs_ch**
- **Code lines:** 15
- **Contained by:** module `iso_varying_string`

**function: l1lt_vs_vs**
- **Code lines:** 14
- **Contained by:** module `iso_varying_string`

**subroutine: load_from_file_vs**
- **Description:** Load a varying string object with the contents of a file (specified by `fileName`).
- **Code lines:** 21
- **Contained by:** module `iso_varying_string`

**subroutine: op_assign_ch_vs**
- **Code lines:** 13
- **Contained by:** module `iso_varying_string`

**subroutine: op_assign_vs_ch**
- **Code lines:** 18
- **Contained by:** module `iso_varying_string`

**function: op_concat_ch_vs**
- **Code lines:** 15
- **Contained by:** module `iso_varying_string`
function: op_concat_vs_ch
  Code lines: 15
  Contained by: module iso_varying_string

function: op_concat_vs_vs
  Code lines: 21
  Contained by: module iso_varying_string

function: op_eq_ch_vs
  Code lines: 15
  Contained by: module iso_varying_string

function: op_eq_vs_ch
  Code lines: 15
  Contained by: module iso_varying_string

function: op_eq_vs_vs
  Code lines: 14
  Contained by: module iso_varying_string

function: op_ge_ch_vs
  Code lines: 15
  Contained by: module iso_varying_string

function: op_ge_vs_ch
  Code lines: 15
  Contained by: module iso_varying_string

function: op_ge_vs_vs
  Code lines: 14
  Contained by: module iso_varying_string

function: op_gt_ch_vs
  Code lines: 15
  Contained by: module iso_varying_string

function: op_gt_vs_ch
  Code lines: 15
  Contained by: module iso_varying_string

function: op_gt_vs_vs
  Code lines: 14
  Contained by: module iso_varying_string

function: op_le_ch_vs
  Code lines: 15
  Contained by: module iso_varying_string

function: op_le_vs_ch
  Code lines: 15
  Contained by: module iso_varying_string

function: op_le_vs_vs
  Code lines: 14
  Contained by: module iso_varying_string

function: op_le_ch_vs
function: op_le_vs_ch
Code lines: 15
Contained by: module iso_varying_string

function: op_le_vs_vs
Code lines: 14
Contained by: module iso_varying_string

function: op_lt_ch_vs
Code lines: 15
Contained by: module iso_varying_string

function: op_lt_vs_ch
Code lines: 15
Contained by: module iso_varying_string

function: op_lt_vs_vs
Code lines: 14
Contained by: module iso_varying_string

function: op_ne_ch_vs
Code lines: 15
Contained by: module iso_varying_string

function: op_ne_vs_ch
Code lines: 15
Contained by: module iso_varying_string

function: op_ne_vs_vs
Code lines: 14
Contained by: module iso_varying_string

interface: operator()
Code lines: 4
Contained by: module iso_varying_string

interface: operator(/)
Code lines: 4
Contained by: module iso_varying_string

interface: operator(/=)
Code lines: 4
Contained by: module iso_varying_string

interface: operator(**)
Code lines: 4
Contained by: module iso_varying_string
interface: put
Code lines: 5
Contained by: module iso_varying_string

subroutine: put_ch
Code lines: 16
Contained by: module iso_varying_string

interface: put_line
Code lines: 5
Contained by: module iso_varying_string

subroutine: put_line_ch
Code lines: 18
Contained by: module iso_varying_string

subroutine: put_line_unit_ch
Code lines: 19
Contained by: module iso_varying_string

subroutine: put_line_unit_vs
Code lines: 15
Contained by: module iso_varying_string

subroutine: put_line_vs
Code lines: 14
Contained by: module iso_varying_string

subroutine: put_unit_ch
Code lines: 19
Contained by: module iso_varying_string

subroutine: put_unit_vs
Code lines: 15
Contained by: module iso_varying_string

subroutine: put_vs
Code lines: 12
Contained by: module iso_varying_string

interface: remove
Code lines: 3
Contained by: module iso_varying_string

function: remove_ch
Code lines: 34
Contained by: module iso_varying_string
function: remove_vs
   Code lines: 15
   Contained by: module iso_varying_string

interface: repeat
   Code lines: 2
   Contained by: module iso_varying_string

function: repeat_
   Code lines: 14
   Contained by: module iso_varying_string

interface: replace
   Code lines: 17
   Contained by: module iso_varying_string

function: replace_ch_ch_auto
   Code lines: 16
   Contained by: module iso_varying_string

function: replace_ch_ch_ch_target
   Code lines: 80
   Contained by: module iso_varying_string

function: replace_ch_ch_fixed
   Code lines: 27
   Contained by: module iso_varying_string

function: replace_ch_ch_vs_target
   Code lines: 19
   Contained by: module iso_varying_string

function: replace_ch_vs_auto
   Code lines: 16
   Contained by: module iso_varying_string

function: replace_ch_vs_ch_target
   Code lines: 19
   Contained by: module iso_varying_string

function: replace_ch_vs_fixed
   Code lines: 17
   Contained by: module iso_varying_string

function: replace_ch_vs_vs_target
   Code lines: 19
   Contained by: module iso_varying_string
19.1. Program units

function: replace_vs_ch_auto
Code lines: 16
Contained by: module iso_varying_string

function: replace_vs_ch_ch_target
Code lines: 19
Contained by: module iso_varying_string

function: replace_vs_ch_fixed
Code lines: 17
Contained by: module iso_varying_string

function: replace_vs_ch_vs_target
Code lines: 19
Contained by: module iso_varying_string

function: replace_vs_vs_auto
Code lines: 16
Contained by: module iso_varying_string

function: replace_vs_vs_ch_target
Code lines: 19
Contained by: module iso_varying_string

function: replace_vs_vs_fixed
Code lines: 17
Contained by: module iso_varying_string

function: replace_vs_vs_vs_target
Code lines: 19
Contained by: module iso_varying_string

interface: scan
Code lines: 4
Contained by: module iso_varying_string

function: scan_ch_vs
Code lines: 23
Contained by: module iso_varying_string

function: scan_vs_ch
Code lines: 23
Contained by: module iso_varying_string

function: scan_vs_vs
Code lines: 23
Contained by: module iso_varying_string
interface: split
  Code lines: 3
  Contained by: module iso_varying_string

subroutine: split_ch
  Code lines: 49
  Contained by: module iso_varying_string

subroutine: split_vs
  Code lines: 21
  Contained by: module iso_varying_string

interface: trim
  Code lines: 2
  Contained by: module iso_varying_string

function: trim_
  Code lines: 13
  Contained by: module iso_varying_string

interface: var_str
  Code lines: 2
  Contained by: module iso_varying_string

function: var_str_
  Code lines: 23
  Contained by: module iso_varying_string

type: varying_string
  Code lines: 16
  Contained by: module iso_varying_string

interface: verify
  Code lines: 4
  Contained by: module iso_varying_string

function: verify_ch_vs
  Code lines: 23
  Contained by: module iso_varying_string

function: verify_vs_ch
  Code lines: 23
  Contained by: module iso_varying_string

function: verify_vs_vs
  Code lines: 23
  Contained by: module iso_varying_string
19.1. Program units

file: math.Bessel_functions.F90
Description: Contains a module which implements calculations of Bessel functions.
Code lines: 65

module: bessel_functions
Description: Implements calculations of Bessel functions.
Code lines: 45
Contained by: file math.Bessel_functions.F90
Modules used: fgs1
Used by: function node_component_disk_-exponential_potential
function node_component_disk_-exponential_rotation_curve_bessel_-factors
function node_component_disk_-exponential_rttn_crv_grdnt_bssl_fctrs
program test_math_special_functions

function: bessel_function_i0
Description: Computes the $I_0$ Bessel function.
Code lines: 7
Contained by: module bessel_functions

function: bessel_function_i1
Description: Computes the $I_1$ Bessel function.
Code lines: 7
Contained by: module bessel_functions

function: bessel_function_k0
Description: Computes the $K_0$ Bessel function.
Code lines: 7
Contained by: module bessel_functions

function: bessel_function_k1
Description: Computes the $K_1$ Bessel function.
Code lines: 7
Contained by: module bessel_functions

file: math.distributions.Gaussian.F90
Description: Contains a module which implements Gaussian distributions.
Code lines: 38

module: math.distributions_gaussian
Description: Implements Gaussian distributions.
Code lines: 18
Contained by: file math.distributions.Gaussian.F90
Used by: function g_1
function g_2
function g_2_integrand_zhang_hui

function: gaussian_distribution
Description: Computes the Gaussian distribution with dispersion $\sigma$ at argument $x$. 
19. Source Code Documentation

Code lines: 9
Contained by: module math_distributions_gaussian
Modules used: numerical_constants_math

file: math.error_function.F90
Description: Contains a module which implements calculations of error functions.
Code lines: 48

module: error_functions
Description: Implements calculations of error functions.
Code lines: 28
Contained by: file math.error_function.F90
Used by: function satellite_dynamical_friction_acceleration_chandrasekhar
         function satellite_tidal_heating_rate_gnedin
         function satellite_tidal_stripping_rate_zentner2005

function: error_function
Description: Computes the error function.
Code lines: 8
Contained by: module error_functions
Modules used: fgsl

function: error_function_complementary
Description: Computes the complementary error function.
Code lines: 8
Contained by: module error_functions
Modules used: fgsl

file: math.exponential_integrals.F90
Description: Contains a module which implements exponential integrals.
Code lines: 47

module: exponential_integrals
Description: Implements exponential integrals.
Code lines: 27
Contained by: file math.exponential_integrals.F90
Modules used: fgsl
Used by: function isothermalkspace
         function nfwkspace
         function nfwkspace
         function isothermalkspace
         program test_math_special_functions

function: cosine_integral
Description: Evaluate the \( \text{Ci}(x) \equiv \int_0^x \frac{dt \cos(t)}{t} \) cosine integral.
Code lines: 7
Contained by: module exponential_integrals

function: sine_integral
Description: Evaluate the \( \text{Si}(x) \equiv \int_0^x \frac{dt \sin(t)}{t} \) sine integral.
19.1. Program units

Code lines: 7  
Contained by: module exponential_integrals

file: math.factorial.F90  
Description: Contains a module which implements calculations of factorials.  
Code lines: 53

module: factorials  
Description: Implements calculations of factorials  
Code lines: 33  
Contained by: file math.factorial.F90  
Used by: function merger_tree_cladistic_ - information_content  
         subroutine meiksin2006apply  
function factorial  
Description: Computes the factorial of argument.  
Code lines: 8  
Contained by: module factorials  
Modules used: fgs1

function: logarithmic_double_factorial  
Description: Computes the natural logarithm of the double factorial, k!!.  
Code lines: 13  
Contained by: module factorials

file: math.gamma_function.F90  
Description: Contains a module which implements calculations of gamma functions.  
Code lines: 111

module: gamma_functions  
Description: Implements calculations of gamma functions.  
Code lines: 91  
Contained by: file math.gamma_function.F90  
Used by: subroutine halo_spin_distribution_ - bett2007_initialize  
         function einastodensityscalefree  
         function einastopotentialscalefree  
function einastorotationnormalization  
function einastorotationnormalization  
function einastorotationnormalization  
function einastorotationnormalization  
function einastorotationnormalization  
subroutine inoue2014apply  
subroutine meiksin2006apply

1227
program test_math_special_functions

function: gamma_function
Description: Computes the gamma function.
Code lines: 7
Contained by: module gamma_functions

function: gamma_function_incomplete
Description: Computes the incomplete gamma function.
Code lines: 8
Contained by: module gamma_functions
Modules used: fgsl

function: gamma_function_incomplete_complementary
Description: Computes the complementary incomplete gamma function.
Code lines: 8
Contained by: module gamma_functions
Modules used: fgsl

function: gamma_function_logarithmic
Description: Computes the logarithm of the gamma function.
Code lines: 8
Contained by: module gamma_functions
Modules used: fgsl

function: inverse_gamma_function_incomplete
Description: Returns the inverse of the incomplete function. That is, it returns $x$ given $Q(a,x)$.
Code lines: 9
Contained by: module gamma_functions

function: inverse_gamma_function_incomplete_complementary
Description: Returns the inverse of the incomplete function. That is, it returns $x$ given $P(a,x)$.
Code lines: 31
Contained by: module gamma_functions
Modules used: galacticus_error incomplete_gamma iso_varying_string

file: math.hypergeometric_functions.F90
Description: Contains a module which implements hypergeometric functions.
Code lines: 57

module: hypergeometric_functions
Description: Implements hypergeometric functions.
Code lines: 37
Contained by: file math.hypergeometric_functions.F90
Modules used: fgsl
Used by: function modified_press_schechter_-subresolution_fraction function mass_distribution_beta_-profile_density_radial_moment

1228
19.1. Program units

subroutine mass_distribution_beta_profile_initialize
function mass_distribution_beta_profile_mass_enc_by_sphere
function mass_distribution_beta_profile_potential
subroutine virial_orbital_parameters_wetzel2010_initialize

function: hypergeometric_1f1
Description: Evaluate the $1F_1(a_1;b_1;x)$ hypergeometric function.
Code lines: 7
Contained by: module hypergeometric_functions

function: hypergeometric_2f1
Description: Evaluate the $2F_1(a_1,a_2;b_1;x)$ hypergeometric function.
Code lines: 17
Contained by: module hypergeometric_functions
Modules used: galacticus_error

file: math.trigonometric_functions.F90
Description: Contains a module which implements trigonometric functions.
Code lines: 75

module: trigonometric_functions
Description: Implements trigonometric functions.
Code lines: 55
Contained by: file math.trigonometric_functions.F90
Modules used: fgsl
Used by: subroutine galacticus_merger_tree_lightcone_geometry_initialize

interface: cosec
Code lines: 3
Contained by: module trigonometric_functions

function: cosecdouble
Description: Implements cosecant for double precision x.
Code lines: 7
Contained by: module trigonometric_functions

function: cosecdoublecomplex
Description: Implements cosecant for double precision complex x.
Code lines: 7
Contained by: module trigonometric_functions

interface: cot
Code lines: 3
Contained by: module trigonometric_functions

function: cotdouble
Description: Implements cotangent for double precision x.
Code lines: 7
Contained by: module trigonometric_functions

function: cotdoublecomplex
Description: Implements cotangent for double precision complex x.
Code lines: 7
Contained by: module trigonometric_functions

file: math.vector.F90
Description: Contains a module which implements calculations of vectors.
Code lines: 78

module: vectors
Description: Implements calculations of vectors.
Code lines: 58
Contained by: file math.vector.F90
Used by: subroutine galacticus_output_analysis_mass_dpndnt_sz_dstrbttins
        subroutine galacticus_merger_tree_lightcone_geometry_initialize
        subroutine geometry_survey_window_functions_caputi_2011_ukidss_uds
        subroutine geometry_survey_window_functions_li_white_2009_sdss
        subroutine geometry_survey_window_functions_martin_2010_alfalfa
        subroutine scan_for_mergers
        subroutine time_until_merging_subresolution
        subroutine galacticusimport
        subroutine node_component_satellite_orbiting_create
        subroutine node_component_satellite_orbiting_rate_compute
        function satellite_dynamical_friction_acceleration_chandrasekhar
        function satellite_tidal_heating_rate_gnedin
        function satellite_tidal_stripping_rate_zentner2005

function: vector_magnitude
Description: Computes the magnitude of vector1.
Code lines: 7
Contained by: module vectors

function: vector_outer_product
Description: Returns the outer product of two vectors.
Code lines: 9
Contained by: module vectors

function: vector_product
Description: Computes the vector product of vector1 and vector2.
Code lines: 10
Contained by: module vectors
19.1. Program units

**function**: vector_self_outer_product  
*Description*: Find the outer product of a vector with itself, returning the result as a rank 2, 3D, symmetric tensor.  
*Code lines*: 16  
*Contained by*: module vectors  
*Modules used*: tensors

**file**: merger_trees.branching_probability.F90  
*Description*: Contains a module which implements calculations of merger tree branching probabilities.  
*Code lines*: 170

**module**: merger_tree_branching  
*Description*: Implements calculations of merger tree branching probabilities.  
*Code lines*: 150  
*Contained by*: file merger_trees.branching_probability.F90  
*Modules used*: iso_varying_string  
*Used by*: subroutine merger_tree_build_do_cole2000

**function**: tree_branch_mass  
*Description*: Return the mass of a progenitor halo in a branch split.  
*Code lines*: 11  
*Contained by*: module merger_tree_branching

**interface**: tree_branch_mass_template  
*Code lines*: 4  
*Contained by*: module merger_tree_branching

**function**: tree_branch_mass_template  
*Code lines*: 2  
*Contained by*: interface tree_branch_mass_template

**subroutine**: tree_branching_initialize  
*Description*: Initializes the tree branching module.  
*Code lines*: 55  
*Contained by*: module merger_tree_branching  
*Modules used*: galacticus_error generalized_press_schechter_branching input_parameters modified_press_schechter_branching

**function**: tree_branching_probability  
*Description*: Return the branching probability per unit $\delta_{\text{crit}}$ for a halo in a merger tree.  
*Code lines*: 11  
*Contained by*: module merger_tree_branching

**interface**: tree_branching_probability_template  
*Code lines*: 4  
*Contained by*: module merger_tree_branching
function: tree_branching_probability_template
Code lines: 2
Contained by: interface tree_branching_probability_template

function: tree_maximum_step
Description: Return the maximum step in $\delta_{\text{crit}}$ allowed for a halo in a merger tree.
Code lines: 11
Contained by: module merger_tree_branching

interface: tree_maximum_step_template
Code lines: 4
Contained by: module merger_tree_branching

function: tree_maximum_step_template
Code lines: 2
Contained by: interface tree_maximum_step_template

function: tree_subresolution_fraction
Description: Return the fraction of mass accreted below the resolution limit per $\delta_{\text{crit}}$ in a halo in a merger tree.
Code lines: 11
Contained by: module merger_tree_branching

interface: tree_subresolution_fraction_template
Code lines: 4
Contained by: module merger_tree_branching

function: tree_subresolution_fraction_template
Code lines: 2
Contained by: interface tree_subresolution_fraction_template

file: merger_trees.branching_probability.generalized_Press_Schechter.F90
Description: Contains a module which implements calculations of branching probabilities in generalized Press-Schechter theory.
Code lines: 389

module: generalized_press_schechter_branching
Description: Implements calculations of branching probabilities in generalized Press-Schechter theory.
Code lines: 369
Contained by: file merger_trees.branching_probability.generalized_Press_Schechter.F90
Modules used: numerical_constants_math power_spectra
Used by: subroutine tree_branching_initialize

function: branching_probability_integrand_generalized
Description: Integrand for the branching probability.
Code lines: 12
Contained by: module generalized_press_schechter_branching
Modules used: iso_c_binding
**subroutine: compute_common_factors**  
*Description:* Precomputes some useful factors that are used in the generalized Press-Schechter branching integrals.  
*Code lines:* 9

*Contained by:* module generalized_press_schechter_branching  
*Modules used:* critical_overdensity

**subroutine: excursion_sets_maximum_sigma_test**  
*Description:* Make a call to excursion set routines with the maximum $\sigma$ that we will use to ensure that they can handle it.  
*Code lines:* 21

*Contained by:* module generalized_press_schechter_branching  
*Modules used:* cosmology_functions excursion_sets_first_crossings

**function: generalized_press_schechter_branch_mass**  
*Description:* Determine the mass of one of the halos to which the given halo branches, given the branching probability, probability. Typically, probabilityFraction is found by multiplying Generalized_Press_Schechter_Branching_Probability() by a random variable drawn in the interval 0–1 if a halo branches. This routine then finds the progenitor mass corresponding to this value.  
*Code lines:* 60

*Contained by:* module generalized_press_schechter_branching  
*Modules used:* galacticus_display galacticus_error iso_varying_string root_finder

**function: generalized_press_schechter_branch_mass_root**

*Code lines:* 12

*Contained by:* module generalized_press_schechter_branching

*Modules used:* iso_c_binding numerical_integration

**subroutine: generalized_press_schechter_branching_initialize**  
*Description:* Initialize the generalized Press-Schechter branching routines.  
*Code lines:* 51

*Contained by:* module generalized_press_schechter_branching

*Modules used:* input_parameters iso_varying_string

**function: generalized_press_schechter_branching_maximum_step**  
*Description:* Return the maximum allowed step in $\delta_{\text{crit}}$ that a halo of mass haloMass at time deltaCritical should be allowed to take.  
*Code lines:* 8

*Contained by:* module generalized_press_schechter_branching

**function: generalized_press_schechter_branching_probability**  
*Description:* Return the probability per unit change in $\delta_{\text{crit}}$ that a halo of mass haloMass at time deltaCritical will undergo a branching to progenitors with mass greater than massResolution.  
*Code lines:* 27

*Contained by:* module generalized_press_schechter_branching

*Modules used:* iso_c_binding numerical_integration
function: generalized_press_schechter_subresolution_fraction
Description: Return the fraction of mass accreted in subresolution halos, i.e. those below massResolution, per unit change in \( \delta_{crit} \) for a halo of mass haloMass at time deltaCritical. The integral is computed numerically.
Code lines: 65
Contained by: module generalized_press_schechter_branching
Modules used: excursion_sets_first_crossings galacticus_display galacticus_error iso_c_binding iso_varying_string merger_tree_branching_modifiers numerical_integration

function: merging_rate
Description: Computes the merging rate of dark matter halos in the generalized Press-Schechter algorithm. This “merging rate” is specifically defined as
\[
\frac{d^2f}{d\ln M_{child}d\delta c} = 2\sigma^2(M_{child}) \left. \frac{d\ln \sigma}{d\ln M} \right|_{M=M_{child}} \frac{dt}{dt} \frac{df_{12}}{dt},
\] (19.21)
where \( df_{12}/dt \) is the excursion set barrier crossing probability per unit time for the effective barrier \( B'(S_{child}|S_{parent},t) \equiv B(S_{child},t - \delta t) - B(S_{parent},t) \) in the limit \( \delta t \to 0 \).
Code lines: 17
Contained by: module generalized_press_schechter_branching
Modules used: excursion_sets_first_crossings merger_tree_branching_modifiers

function: progenitor_mass_function
Description: Progenitor mass function from Press-Schechter.
Code lines: 7
Contained by: module generalized_press_schechter_branching

function: subresolution_fraction_integrand_generalized
Description: Integrand for the subresolution fraction.
Code lines: 16
Contained by: module generalized_press_schechter_branching
Modules used: iso_c_binding

file: merger_trees.branching_probability.modified_Press_Schechter.F90
Description: Contains a module which implements calculations of branching probabilities in modified Press-Schechter theory.
Code lines: 305

module: modified_press_schechter_branching
Description: Implements calculations of branching probabilities in modified Press-Schechter theory.
Code lines: 285
Contained by: file merger_trees.branching_probability.modified_Press_Schechter.F90
Modules used: numerical_constants_math power_spectra
Used by: subroutine tree_branching_initialize

function: branching_probability_integrand
19.1. Program units

**Description:** Integrand for the branching probability.

**Code lines:** 12

**Contained by:** module `modified_press_schechter_branching`

**Modules used:** `iso_c_binding`

**subroutine: compute_common_factors**

**Description:** Precomputes some useful factors that are used in the modified Press-Schechter branching integrals.

**Code lines:** 8

**Contained by:** module `modified_press_schechter_branching`

**function: merging_rate**

**Description:** Merging rate from Press-Schechter. The constant factor of $\sqrt{2/\pi}$ not included here—instead it is included in a multiplicative prefactor by which integrals over this function are multiplied.

**Code lines:** 15

**Contained by:** module `modified_press_schechter_branching`

**function: modification_function**

**Description:** Empirical modification of the progenitor mass function from Parkinson et al. [2008]. The constant factors of $G_0(\delta_p/\sigma_p)^{\gamma_2}$ and $1/\sigma_p^{\gamma_1}$ are not included here—instead they are included in a multiplicative prefactor by which integrals over this function are multiplied.

**Code lines:** 11

**Contained by:** module `modified_press_schechter_branching`

**function: modified_press_schechter_branch_mass**

**Description:** Determine the mass of one of the halos to which the given halo branches, given the branching probability, \texttt{probability}. Typically, \texttt{probabilityFraction} is found by multiplying Modified_Press_Schechter_Branching_Probability() by a random variable drawn in the interval 0–1 if a halo branches. This routine then finds the progenitor mass corresponding to this value.

**Code lines:** 33

**Contained by:** module `modified_press_schechter_branching`

**Modules used:** `root_finder`

**function: modified_press_schechter_branch_mass_root**

**Code lines:** 12

**Contained by:** module `modified_press_schechter_branching`

**Modules used:** `iso_c_binding`, `numerical_integration`

**subroutine: modified_press_schechter_branching_initialize**

**Description:** Initialize the modified Press-Schechter branching routines.

**Code lines:** 64

**Contained by:** module `modified_press_schechter_branching`

**Modules used:** `input_parameters`, `iso_varying_string`

**function: modified_press_schechter_branching_maximum_step**

**Description:** Return the maximum allowed step in $\delta_{crit}$ that a halo of mass \texttt{haloMass} at time \texttt{deltaCritical} should be allowed to take.

**Code lines:** 17
19. Source Code Documentation

**Contained by:** module *modified_press_schechter_branching*

**function:** *modified_press_schechter_branching_probability*

**Description:** Return the probability per unit change in $\delta_{\text{crit}}$ that a halo of mass $\text{haloMass}$ at time $\delta_{\text{critical}}$ will undergo a branching to progenitors with mass greater than $\text{massResolution}$.

**Code lines:** 26

**Contained by:** module *modified_press_schechter_branching*

**Modules used:** iso_c_binding, numerical_integration

**function:** *modified_press_schechter_subresolution_fraction*

**Description:** Return the fraction of mass accreted in subresolution halos, i.e. those below $\text{massResolution}$, per unit change in $\delta_{\text{crit}}$ for a halo of mass $\text{haloMass}$ at time $\delta_{\text{critical}}$. The integral is computed analytically in terms of the $2F_1$ hypergeometric function.

**Code lines:** 28

**Contained by:** module *modified_press_schechter_branching*

**Modules used:** hypergeometric_functions

**function:** *progenitor_mass_function*

**Description:** Progenitor mass function from Press-Schechter. The constant factor of the parent halo mass is not included here—instead it is included in a multiplicative prefactor by which integrals over this function are multiplied.

**Code lines:** 9

**Contained by:** module *modified_press_schechter_branching*

**file:** merger_trees.branching_probability.modifier.F90

**Description:** Contains a module which implements modifiers for merger tree branching probabilities.

**Code lines:** 111

**module:** merger_tree_branching_modifiers

**Description:** Implements modifiers for merger tree branching probabilities.

**Code lines:** 91

**Contained by:** file merger_trees.branching_probability.modifier.F90

**Modules used:** iso_varying_string

**Used by:** function generalized_press_schechter_, subresolution_fraction

**function:** merger_tree_branching_modifier

**Description:** Return a modifier for merger tree branching probabilities.

**Code lines:** 11

**Contained by:** module *merger_tree_branching_modifiers*

**subroutine:** tree_branching_modifiers_initialize

**Description:** Initializes the tree branching modifier module.

**Code lines:** 53

**Contained by:** module *merger_tree_branching_modifiers*

**Modules used:** galacticus_error, input_parameters, merger_tree_branching_modifiers_null, merger_tree_branching_modifiers_parkinson
file: merger_trees.branching_probability.modifier.Parkinson.F90
Description: Contains a module which implements the Parkinson et al. [2008] modifier of merger tree branch-
ing rates.
Code lines: 100

module: merger_tree_branching_modifiers_parkinson
Description: Implements the Parkinson et al. [2008] modifier of merger tree branching rates.
Code lines: 80
Contained by: file merger_trees.branching_probability.modifier.Parkinson.F90
Used by: subroutine tree_branching_modifiers_-_initialize

function: merger_tree_branching_modifier_parkinson
Description: Returns a modifier for merger tree branching rates using the Parkinson et al. [2008] algo-

rithm.
Code lines: 17
Contained by: module merger_tree_branching_modifiers_parkinson

subroutine: merger_tree_branching_modifiers_parkinson_initialize
Description: Initialize the null modifier method for merger tree branching rates.
Code lines: 45
Contained by: module merger_tree_branching_modifiers_parkinson
Modules used: input_parameters iso_varying_string

file: merger_trees.branching_probability.modifier.null.F90
Description: Contains a module which implements calculations of branching probabilties in modified
Press-Schechter theory.
Code lines: 51

module: merger_tree_branching_modifiers_null
Description: Implements a null modifier of merger tree branching rates.
Code lines: 31
Contained by: file merger_trees.branching_probability.modifier.null.F90
Used by: subroutine tree_branching_modifiers_-_initialize

function: merger_tree_branching_modifier_null
Description: Returns a null (multiplicative) modifier for merger tree branching rates.
Code lines: 7
Contained by: module merger_tree_branching_modifiers_null

subroutine: merger_tree_branching_modifiers_null_initialize
Description: Initialize the null modifier method for merger tree branching rates.
Code lines: 9
Contained by: module merger_tree_branching_modifiers_null
Modules used: iso_varying_string

file: merger_trees.conditional_mass_functions.F90
19. Source Code Documentation

**Description:** Contains a module which computes conditional mass functions in merger trees.
**Code lines:** 467

**module:** merger_trees_conditional_mass_function

**Description:** Computes conditional mass functions in merger trees.
**Code lines:** 447
**Contained by:** file merger_trees.conditional_mass_functions.F90
**Used by:** subroutine galacticus_output_close_file  function galacticus_task_evolve_tree

**subroutine:** merger_tree_conditional_mass_function

**Description:** Compute conditional mass function on thisTree.
**Code lines:** 360
**Contained by:** module merger_trees_conditional_mass_function
**Modules used:** cosmology_functions  galacticus_error
galacticus_nodes  input_parameters
memory_management  numerical_comparison
numerical_ranges

**subroutine:** merger_tree_conditional_mass_function_output

**Description:** Outputs conditional mass function.
**Code lines:** 46
**Contained by:** module merger_trees_conditional_mass_function
**Modules used:** galacticus_hdf5  io_hdf5
iso_varying_string  numerical_constants_astronomical

**file:** merger_trees.construct.F90

**Description:** Contains a module which constructs/destructs merger trees.
**Code lines:** 129

**module:** merger_tree_construction

**Description:** Constructs/destructs merger trees.
**Code lines:** 109
**Contained by:** file merger_trees.construct.F90
**Modules used:** galacticus_nodes  iso_varying_string
**Used by:** subroutine get_tree

**function:** merger_tree_create

**Description:** Creates a merger tree.
**Code lines:** 82
**Contained by:** module merger_tree_construction
**Modules used:** galacticus_error  input_parameters
memory_management  merger_tree_build
merger_tree_read  merger_tree_smooth_accretion
merger_trees_construct_fully_specified  merger_trees_state_store

**file:** merger_trees.construct.build.Cole2000.F90

**Description:** Contains a module which implements building of merger trees using the algorithm of Cole et al. [2000].
**Code lines:** 323
module: merger_tree_build_cole2000
Description: Implements building of merger trees using the algorithm of Cole et al. [2000].
Code lines: 303
Modules used: fgsl
Used by: subroutine galacticus_state_retrieve subroutine galacticus_state_snapshot
          subroutine galacticus_state_store subroutine merger_tree_build_initialize

subroutine: merger_tree_build_cole2000_initialize
Description: Initializes the Cole et al. [2000] merger tree building module.
Code lines: 65
Contained by: module merger_tree_build_cole2000
Modules used: cosmology_functions input_parameters
               iso_varying_string

subroutine: merger_tree_build_cole2000_snapshot
Description: Store a snapshot of the random number generator internal state.
Code lines: 11
Contained by: module merger_tree_build_cole2000
Modules used: pseudo_random

subroutine: merger_tree_build_cole2000_state_retrieve
Description: Write the stored snapshot of the random number state to file.
Code lines: 10
Contained by: module merger_tree_build_cole2000
Modules used: pseudo_random

subroutine: merger_tree_build_cole2000_state_store
Description: Write the stored snapshot of the random number state to file.
Code lines: 10
Contained by: module merger_tree_build_cole2000
Modules used: pseudo_random

subroutine: merger_tree_build_do_cole2000
Description: Build a merger tree.
Code lines: 165
Contained by: module merger_tree_build_cole2000
Modules used: critical_overdensity galacticus_error
galacticus_nodes kind_numbers
merger_tree_branching merger_trees_build_mass_resolution
pseudo_random

file: merger_trees.construct.build.F90
Description: Contains a module which implements building of merger trees after drawing masses at
             random from a mass function.
Code lines: 461

module: merger_tree_build
19. Source Code Documentation

Description: Implements building of merger trees after drawing masses at random from a mass function.
Code lines: 441
Contained by: file merger_trees.construct.build.F90
Modules used: galacticus_nodes iso_varying_string
Used by: function merger_tree_create

function: mass_function_sampling_integrand
Description: The integrand over the mass function sampling density function.
Code lines: 11
Contained by: module merger_tree_build
Modules used: merger_trees_mass_function_sampling

subroutine: merger_tree_build_do
Description: Build a merger tree.
Code lines: 55
Contained by: module merger_tree_build
Modules used: galacticus_state kind_numbers string_handling

subroutine: merger_tree_build_initialize
Description: Initializes the merger tree building module.
Code lines: 333
Contained by: module merger_tree_build
Modules used: cosmology_functions fgs1 fox_dom galacticus_display galacticus_error halo_mass_function input_parameters io_hdf5 io_xml iso_c_binding memory_management merger_tree_build_cole2000 numerical_integration numerical_interpolation numerical_ranges pseudo_random quasi_random sort

file: merger_trees.construct.build.mass_resolution.F90
Description: Contains a module which implements calculations of merger tree building mass resolutions.
Code lines: 107

module: merger_trees_build_mass_resolution
Description: Implements calculations of merger tree branching probabilities.
Code lines: 87
Contained by: file merger_trees.construct.build.mass_resolution.F90
Modules used: isovarying_string
Used by: subroutine merger_tree_build_do_cole2000

function: merger_tree_build_mass_resolution
Description: Return the mass resolution to use when building thisTree.
Code lines: 12
19.1. Program units

**Contained by:** module `merger_trees_build_mass_resolution`

**Modules used:** `galacticus_nodes`

**subroutine:** `merger_tree_build_mass_resolution_initialize`

**Description:** Initializes the merger tree mass resolution module.

**Code lines:** 56

**Contained by:** module `merger_trees_build_mass_resolution`

**Modules used:**
- `galacticuserror`
- `input_parameters`
- `merger_trees_build_mass_resolution_fixed`
- `merger_trees_build_mass_resolution_scaled`

**file:** `merger_trees.construct.build.mass_resolution.fixed.F90`

**Description:** Contains a module which implements a fixed mass resolution for building merger trees.

**Code lines:** 69

**module:** `merger_trees_build_mass_resolution_fixed`

**Description:** Implements a fixed mass resolution for building merger trees.

**Code lines:** 49

**Contained by:** file `merger_trees.construct.build.mass_resolution.fixed.F90`

**Used by:** subroutine `merger_tree_build_mass_resolution_initialize`

**function:** `merger_tree_build_mass_resolution_fixed`

**Description:** Returns a fixed mass resolution to use when building merger trees.

**Code lines:** 8

**Contained by:** module `merger_trees_build_mass_resolution_fixed`

**Modules used:** `galacticus_nodes`

**subroutine:** `merger_trees_build_mass_resolution_fixed_initialize`

**Description:** Initialize the modified Press-Schechter branching routines.

**Code lines:** 23

**Contained by:** module `merger_trees_build_mass_resolution_fixed`

**Modules used:** `input_parameters`

**file:** `merger_trees.construct.build.mass_resolution.scaled.F90`

**Description:** Contains a module which implements a scaled mass resolution for building merger trees.

**Code lines:** 84

**module:** `merger_trees_build_mass_resolution_scaled`

**Description:** Implements a scaled mass resolution for building merger trees.

**Code lines:** 64

**Contained by:** file `merger_trees.construct.build.mass_resolution.scaled.F90`

**Used by:** subroutine `merger_tree_build_mass_resolution_scaled.initialize`

**function:** `merger_tree_build_mass_resolution_scaled`

**Description:** Returns a scaled mass resolution to use when building merger trees.

**Code lines:** 12

**Contained by:** module `merger_trees_build_mass_resolution_scaled`
19. Source Code Documentation

Modules used: galacticus_nodes

**subroutine:** merger_trees_build_mass_resolution_scaled_initialize
**Description:** Initialize the modified Press-Schechter branching routines.
**Code lines:** 34
**Contained by:** module merger_trees_build_mass_resolution_scaled
**Modules used:** input_parameters iso_varying_string

**file:** merger_trees.construct.fully_specified.F90
**Description:** Contains a module which implements building of merger trees using a fully-specified description read from file.
**Code lines:** 200

**module:** merger_trees_construct_fully_specified
**Description:** Implements building of merger trees using a fully-specified description read from file.
**Code lines:** 180
**Contained by:** file merger_trees.construct.fully_specified.F90
**Modules used:** iso_varying_string
**Used by:** function merger_tree_create

**subroutine:** merger_tree_construct_fully_specified
**Description:** Construct a fully-specified merger tree.
**Code lines:** 86
**Contained by:** module merger_trees_construct_fully_specified
**Modules used:** fox_dom galacticus_display galacticus_error galacticus_nodes kind_numbers memory_management

**subroutine:** merger_tree_construct_fully_specified_initialize
**Description:** Initializes the merger tree construction “fully-specified” module.
**Code lines:** 24
**Contained by:** module merger_trees_construct_fully_specified
**Modules used:** input_parameters

**function:** node_definition_index
**Description:** Extract and return an index from a node definition as used when constructing fully-specified merger trees.
**Code lines:** 23
**Contained by:** module merger_trees_construct_fully_specified
**Modules used:** fox_dom galacticus_error kind_numbers

**function:** node_lookup
**Description:** Find the position of a node in the nodeArray array given its indexValue.
**Code lines:** 21
**Contained by:** module merger_trees_construct_fully_specified
**Modules used:** galacticus_error galacticus_nodes kind_numbers
19.1. Program units

file: merger_trees.construct.mass_function_sampling.F90
Description: Contains a module which implements methods for sampling the halo mass function when constructing merger trees.
Code lines: 116

module: merger_trees_mass_function_sampling
Description: Implements methods for sampling the halo mass function when constructing merger trees.
Code lines: 96
Contained by: file merger_trees.construct.mass_function_sampling.F90
Modules used: iso_varying_string
Used by: function mass_function_sampling_- integrand

function: merger_tree_construct_mass_function_sampling
Description: Returns the sampling rate for merger trees of the given mass, per decade of halo mass.
Code lines: 12
Contained by: module merger_trees_mass_function_sampling

subroutine: merger_trees_mass_function_sampling_initialize
Description: Initialize the halo mass function sampling module.
Code lines: 57
Contained by: module merger_trees_mass_function_sampling
Modules used: galacticus_error input_parameters
merger_trees_mass_function_sampling_- gaussian
merger_trees_mass_function_sampling_- halo_mf
merger_trees_mass_function_sampling_- power_law
merger_trees_mass_function_sampling_- stellar_mf

file: merger_trees.construct.mass_function_sampling.gaussian.F90
Description: Contains a module which implements halo mass function sampling using a Gaussian in halo mass.
Code lines: 83

module: merger_trees_mass_function_sampling_gaussian
Description: Implements halo mass function sampling using a Gaussian in halo mass.
Code lines: 63
Contained by: file merger_trees.construct.mass_function_sampling.gaussian.F90
Used by: subroutine merger_trees_mass_function_sampling_initialize

function: merger_tree_construct_mass_function_sampling_gaussian
Description: Computes the halo mass function sampling rate using a Gaussian distribution.
Code lines: 11
Contained by: module merger_trees_mass_function_sampling_gaussian

subroutine: merger_trees_mass_function_sampling_gaussian_initialize
Description: Initializes the “gaussian” halo mass function sampling method.
Code lines: 35
19. Source Code Documentation

**Contained by:** module `merger_trees_mass_function_sampling_gaussian`
**Modules used:** `input_parameters` `iso_varying_string`

**file:** `merger_trees.construct.mass_function_sampling.halo_mass_function.F90`
**Description:** Contains a module which implements halo mass function sampling proportional to halo abundance (i.e. a volume-limited sample of halos).
**Code lines:** 81

**module:** `merger_trees_mass_function_sampling_halo_mf`
**Description:** Implements halo mass function sampling proportional to halo abundance (i.e. a volume-limited sample of halos).
**Code lines:** 61
**Contained by:** file `merger_trees.construct.mass_function_sampling.halo_mass_function.F90`
**Used by:** subroutine `merger_trees_mass_function_sampling_initialize`

**function:** `merger_tree_construct_mass_function_sampling_halo_mf`
**Description:** Computes the halo mass function sampling rate using a volume-limited sampling.
**Code lines:** 10
**Contained by:** module `merger_trees_mass_function_sampling_halo_mf`
**Modules used:** `halo_mass_function`

**subroutine:** `merger_trees_mass_function_sampling_halo_mf_initialize`
**Description:** Initializes the “haloMassFunction” halo mass function sampling method.
**Code lines:** 34
**Contained by:** module `merger_trees_mass_function_sampling_halo_mf`
**Modules used:** `input_parameters` `iso_varying_string`

**file:** `merger_trees.construct.mass_function_sampling.power_law.F90`
**Description:** Contains a module which implements halo mass function sampling using a power-law in halo mass.
**Code lines:** 74

**module:** `merger_trees_mass_function_sampling_power_law`
**Description:** Implements halo mass function sampling using a power-law in halo mass.
**Code lines:** 54
**Contained by:** file `merger_trees.construct.mass_function_sampling.power_law.F90`
**Used by:** subroutine `merger_trees_mass_function_sampling_power_law_initialize`

**function:** `merger_tree_construct_mass_function_sampling_power_law`
**Description:** Computes the halo mass function sampling rate using a power-law distribution.
**Code lines:** 12
**Contained by:** module `merger_trees_mass_function_sampling_power_law`

**subroutine:** `merger_trees_mass_function_sampling_power_law_initialize`
**Description:** Initializes the “powerLaw” halo mass function sampling method.
**Code lines:** 25
**Contained by:** module `merger_trees_mass_function_sampling_power_law`
19.1. Program units

**Modules used:**

- input_parameters
- iso_varying_string

**file:** merger_trees.construct.mass_function_sampling.stellar_mass_function.F90

- **Description:** Contains a module which implements halo mass function sampling optimized to minimize variance in the model stellar mass function.
- **Code lines:** 194

**module:** merger_trees_mass_function_sampling_stellar_mf

- **Description:** Implements halo mass function sampling optimized to minimize variance in the model stellar mass function.
- **Code lines:** 174
- **Contained by:** file merger_trees.construct.mass_function_sampling.stellar_mass_function.F90
- **Used by:** subroutine merger_trees_mass_function_sampling.initialize

**function:** merger_tree_construct_mass_function_sampling_stellar_mf

- **Description:** Computes the halo mass function sampling rate using a power-law distribution.
- **Code lines:** 34
- **Contained by:** module merger_trees_mass_function_sampling_stellar_mf
- **Modules used:**
  - fgsl
  - galacticus_meta_compute_times
  - halo_mass_function
  - iso_c_binding
  - numerical_integration

**subroutine:** merger_trees_mass_function_sampling_stellar_mf_initialize

- **Description:** Initializes the “stellarMassFunction” halo mass function sampling method.
- **Code lines:** 94
- **Contained by:** module merger_trees_mass_function_sampling_stellar_mf
- **Modules used:**
  - input_parameters
  - iso_varying_string

**function:** xi_integrand

- **Description:** The integrand appearing in the ξ function.
- **Code lines:** 24
- **Contained by:** module merger_trees_mass_function_sampling_stellar_mf
- **Modules used:**
  - conditional_mass_functions
  - iso_c_binding

**file:** merger_trees.construct.read.F90

- **Description:** Contains a module which implements reading of merger trees from a file.
- **Code lines:** 2670

**module:** merger_tree_read

- **Description:** Implements reading of merger trees from a file.
- **Code lines:** 2650
- **Contained by:** file merger_trees.construct.read.F90
- **Modules used:**
  - galacticus_nodes
  - iso_varying_string
  - kind_numbers
  - merger_tree_read_importers
- **Used by:** function merger_tree_create

**subroutine:** assign_isolated_node_indices
19. Source Code Documentation

**Description:** Assign to each node the number of the corresponding isolated node.

**Code lines:** 40

**Contained by:** module `merger_tree_read`

**subroutine:** `assign_mergers`

**Description:** Assign pointers to merge targets.

**Code lines:** 41

**Contained by:** module `merger_tree_read`

**Modules used:**
- `galacticus_error`
- `string_handling`

**subroutine:** `assign_particle_counts`

**Description:** Assign particle counts to nodes.

**Code lines:** 20

**Contained by:** module `merger_tree_read`

**subroutine:** `assign_scale_radii`

**Description:** Assign scale radii to nodes.

**Code lines:** 142

**Contained by:** module `merger_tree_read`

**Modules used:**
- `dark_matter_halo_scales`
- `dark_matter_profile_scales`
- `dark_matter_profiles_concentration`
- `fallbackscalemethod`
- `galacticus_display`
- `galacticus_error`
- `input_parameters`
- `root_finder`

**subroutine:** `assign_spin_parameters`

**Description:** Assign spin parameters to nodes.

**Code lines:** 55

**Contained by:** module `merger_tree_read`

**Modules used:**
- `dark_matter_profiles`
- `galacticus_error`
- `halo_spin_distributions`
- `numerical_constants_physical`

**subroutine:** `assign_uniqueids_to_clones`

**Description:** Assign new uniqueID values to any cloned nodes inserted into the trees.

**Code lines:** 12

**Contained by:** module `merger_tree_read`

**subroutine:** `assign_velocity_dispersions`

**Description:** Assign velocity dispersions to nodes.

**Code lines:** 20

**Contained by:** module `merger_tree_read`

**subroutine:** `assign_velocity_maxima`

**Description:** Assign velocity maxima to nodes.

**Code lines:** 20

**Contained by:** module `merger_tree_read`

**subroutine:** `build_child_and_sibling_links`

**Description:** Build child and sibling links between nodes.

**Code lines:** 62
19.1. Program units

**Contained by:** module `merger_tree_read`
**Modules used:** `memory_management`

**subroutine: build_descendent_pointers**
**Description:** Builds pointers from each node to its descendent node.
**Code lines:** 44
**Contained by:** module `merger_tree_read`
**Modules used:** `galacticus_display` `galacticus_error`
**Code lines:** 44

**subroutine: build_isolated_parent_pointers**
**Description:** Create parent pointer links between isolated nodes and assign times and masses to those nodes.
**Code lines:** 82
**Contained by:** module `merger_tree_read`
**Modules used:** `galacticus_error` `string_handling`

**subroutine: build_parent_pointers**
**Description:** Build pointers to node parents.
**Code lines:** 42
**Contained by:** module `merger_tree_read`
**Modules used:** `galacticus_error` `string_handling`

**subroutine: build_subhalo_mass_histories**
**Description:** Build and attached bound mass histories to subhalos.
**Code lines:** 135
**Contained by:** module `merger_tree_read`
**Modules used:** `cosmology_functions` `galacticus_error`
**Code lines:** 135

**subroutine: create_branch_jump_event**
**Description:** Create a matched-pair of branch jump events in the given nodes.
**Code lines:** 18
**Contained by:** module `merger_tree_read`
**Modules used:** `node_branch_jumps`

**subroutine: create_node_array**
**Description:** Create an array of standard nodes and associated structures.
**Code lines:** 53
**Contained by:** module `merger_tree_read`
**Modules used:** `memory_management`

**subroutine: create_node_indices**
**Description:** Create a sorted list of node indices with an index into the original array.
**Code lines:** 34
**Contained by:** module `merger_tree_read`
**Modules used:** `galacticus_display` `memory_management`
**Code lines:** 34

sort `string_handling`
function: descendant_node_sort_index
  Description: Return the sort index of the given descendantIndex.
  Code lines: 9
  Contained by: module merger_tree_read
  Modules used: arrays_search

subroutine: destroy_node_indices
  Description: Destroy the sorted list of node indices.
  Code lines: 10
  Contained by: module merger_tree_read
  Modules used: memory_management

subroutine: dump_tree
  Description: Dumps the tree structure to a file in a format suitable for processing with DOT.
  Code lines: 64
  Contained by: module merger_tree_read

subroutine: enforce_subhalo_status
  Description: Ensure that any node which was once a subhalo remains a subhalo.
  Code lines: 69
  Contained by: module merger_tree_read
  Modules used: galacticus_error string_handling

function: half_mass_radius_root
  Description: Function used to find scale radius of dark matter halos given their half-mass radius.
  Code lines: 14
  Contained by: module merger_tree_read
  Modules used: dark_matter_profiles

function: is_subhalo_subhalo_merger
  Description: Returns true if thisNode undergoes a subhalo-subhalo merger.
  Code lines: 24
  Contained by: module merger_tree_read

function: last_host_descendent
  Description: Return a pointer to the last descendent that can be reached from thisNode when descending through hosts.
  Code lines: 11
  Contained by: module merger_tree_read

subroutine: merger_tree_read_do
  Description: Read a merger tree from file.
  Code lines: 268
  Contained by: module merger_tree_read
  Modules used: array_utilities arrays_search
  cosmo
 ology_functions galacticus_error
  galacticus_state memory_management
  numerical_comparison string_handling
19.1. Program units

vectors

**subroutine: merger_tree_read_initilize**
*Description:* Initializes the merger tree reading module.
*Code lines:* 407
*Contained by:* module merger_tree_read
*Modules used:* cosmology_parameters cosmology_functions
galacticus_error galacticus_output_times
input_parameters iso_c_binding
memory_management numerical_constants
numerical_constants_boolean

**function: node_location**
*Description:* Return the location in the original array of the given nodeIndex.
*Code lines:* 15
*Contained by:* module merger_tree_read
*Modules used:* arrays_search

**function: orbit_construct**
*Description:* Construct a Keplerian orbit given body masses, positions, and relative velocities.
*Code lines:* 15
*Contained by:* module merger_tree_read
*Modules used:* kepler_orbits vectors

**subroutine: phase_space_position_realize**
*Description:* Modify relative positions and velocities to account for both any periodicity of the simulated volume, and for Hubble flow.
*Code lines:* 22
*Contained by:* module merger_tree_read
*Modules used:* cosmology_functions numerical_constants_boolean

**type: progenitoriterator**
*Code lines:* 43
*Contained by:* module merger_tree_read

**function: progenitoriteratorcurrent**
*Description:* Return a pointer to the current progenitor in a progenitor iterator object.
*Code lines:* 9
*Contained by:* module merger_tree_read

**subroutine: progenitoriteratordescendentset**
*Description:* Initialize a progenitor iterator object by storing the index of the target node and finding the location of the first progenitor (if any).
*Code lines:* 23
*Contained by:* module merger_tree_read

**function: progenitoriteratorexist**
*Description:* Return true if progenitors exist, false otherwise.
*Code lines:* 7
19. Source Code Documentation

function: progenitoriteratorindex
Description: Return the node index of the current progenitor in a progenitor iterator object.
Code lines: 9
Contained by: module merger_tree_read

function: progenitoriteratornext
Description: Move to the next progenitor using a progenitor iterator object, returning true if the next progenitor exists, false if it does not.
Code lines: 25
Contained by: module merger_tree_read

subroutine: scan_for_branch_jumps
Description: Search for subhalos which move between branches/trees.
Code lines: 136
Contained by: module merger_tree_read
Modules used: galacticus_display iso_varying_string string_handling

subroutine: scan_for_mergers
Description: Scan for and record mergers between nodes.
Code lines: 261
Contained by: module merger_tree_read
Modules used: dark_matter_halo_scales galacticus_error kepler_orbits string_handling vectors virial_orbits

subroutine: scan_for_subhalo_promotions
Description: Scan for cases where a subhalo stops being a subhalo and so must be promoted.
Code lines: 59
Contained by: module merger_tree_read
Modules used: node_subhalo_promotions

subroutine: time_until_merging_subresolution
Description: Compute the additional time until merging after a subhalo is lost from the tree (presumably due to limited resolution).
Code lines: 127
Contained by: module merger_tree_read
Modules used: galacticus_display galacticus_error input_parameters kepler_orbits satellite_merging_timescales string_handling vectors

subroutine: validate_isolated_halos
Description: Ensure that nodes have valid primary progenitors.
Code lines: 42
Contained by: module merger_tree_read
file: merger_trees.construct.read.importer.F90
Description: Contains a module which provides an object that implements importing of merger trees from file.
Code lines: 3716

module: merger_tree_read_importers
Description: Provides an object that implements importing of merger trees from file. An implementation of the merger tree importer class for GALACTICUS format merger tree files. An implementation of the merger tree importer class for GALACTICUS format merger tree files.
Code lines: 3696
Contained by: file merger_trees.construct.read.importer.F90
Modules used: galacticus_nodes io_hdf5
iso_c_binding iso_varying_string
kind_numbers stateful_types
Used by: module merger_tree_read

function: galacticusangularmomenta3davailable
Description: Return true if angular momenta vectors are available.
Code lines: 7
Contained by: module merger_tree_read_importers

function: galacticusangularmomentaavailable
Description: Return true if angular momenta are available.
Code lines: 7
Contained by: module merger_tree_read_importers

function: galacticusangularmomentaincludesubhalos
Description: Return a Boolean specifying whether or not the halo momenta include the contribution from subhalos.
Code lines: 24
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

subroutine: galacticusclose
Description: Validate a GALACTICUS format merger tree file.
Code lines: 11
Contained by: module merger_tree_read_importers

function: galacticuscubelength
Description: Return the length of the simulation cube.
Code lines: 37
Contained by: module merger_tree_read_importers
Modules used: galacticus_error numerical_constants_astronomical
numerical_constants_boolean

function: galacticusdefaultconstructor
Description: Default constructor for the GALACTICUS format merger tree importer.
Code lines: 46
19. Source Code Documentation

```
Contained by:  module merger_tree_read_importers
Modules used:  input_parameters

subroutine: galacticusDestructor
Description:  Destructor for the GALACTICUS format merger tree importer class.
Code lines:    7
Contained by:  module merger_tree_read_importers

subroutine: galacticusimport
Description:  Import the ith merger tree.
Code lines:    190
Contained by:  module merger_tree_read_importers
Modules used:  cosmology_functions  galacticus_display
galacticus_error
memory_management
numerical_constants_astronomical
numerical_constants_prefixes
vectors

function: galacticusmassesincludesubhalos
Description:  Return a Boolean specifying whether or not the halo masses include the contribution from subhalos.
Code lines:    20
Contained by:  module merger_tree_read_importers
Modules used:  galacticus_error

function: galacticusnodecount
Description:  Return a count of the number of nodes in the ith tree.
Code lines:    10
Contained by:  module merger_tree_read_importers

subroutine: galacticusopen
Description:  Validate a GALACTICUS format merger tree file.
Code lines:    180
Contained by:  module merger_tree_read_importers
Modules used:  cosmology_parameters  galacticus_display
galacticus_error
numerical_comparison
power_spectra

function: galacticusparticlecountavailable
Description:  Return true if particle counts are available.
Code lines:    9
Contained by:  module merger_tree_read_importers

function: galacticuspositionsareperiodic
Description:  Return a Boolean integer specifying whether or not positions are periodic.
Code lines:    18
Contained by:  module merger_tree_read_importers
Modules used:  numerical_constants_boolean

function: galacticuspositionsavailable
```

19.1. Program units

**Description:** Return true if positions and/or velocities are available.
**Code lines:** 12
**Contained by:** module `merger_tree_read_importers`

**function: galacticusscaleradiiavailable**
**Description:** Return true if scale radii are available.
**Code lines:** 9
**Contained by:** module `merger_tree_read_importers`

**function: galacticusspin3davailable**
**Description:** Return true if spins vectors are available.
**Code lines:** 7
**Contained by:** module `merger_tree_read_importers`

**function: galacticusspinavailable**
**Description:** Return true if spins are available.
**Code lines:** 7
**Contained by:** module `merger_tree_read_importers`

**subroutine: galacticussubhalotrace**
**Description:** Returns a trace of subhalo position/velocity.
**Code lines:** 42
**Contained by:** module `merger_tree_read_importers`
**Modules used:**
- `cosmology_functions`
- `galacticus_error`
- `numerical_constants_astronomical`

**function: galacticussubhalotracecount**
**Description:** Returns the length of a subhalo trace.
**Code lines:** 15
**Contained by:** module `merger_tree_read_importers`
**Modules used:** `galacticus_error`

**function: galacticustreecount**
**Description:** Return a count of the number of trees available.
**Code lines:** 9
**Contained by:** module `merger_tree_read_importers`

**function: galacticustreeindex**
**Description:** Return the index of the \(i^{th}\) tree.
**Code lines:** 9
**Contained by:** module `merger_tree_read_importers`

**subroutine: galacticustreeindicesread**
**Description:** Read the tree indices.
**Code lines:** 104
**Contained by:** module `merger_tree_read_importers`
**Modules used:**
- `cosmology_functions`
- `galacticus_error`
- `halo_mass_function`
- `hdf5`
function: galacticustreesareselfcontained
Description: Return a Boolean integer specifying whether or not the trees are self-contained.
Code lines: 18
Contained by: module merger_tree_read_importers
Modules used: numerical_constants_boolean

function: galacticustreeshavesubhalos
Description: Return a Boolean integer specifying whether or not the trees have subhalos.
Code lines: 18
Contained by: module merger_tree_read_importers
Modules used: numerical_constants_boolean

function: galacticustreeweight
Description: Return the weight to assign to trees.
Code lines: 33
Contained by: module merger_tree_read_importers
Modules used: cosmology_functions galacticus_error numerical_constants_astronomical numerical_constants_boolean

function: galacticusvelocitiesincludehubbleflow
Description: Return a Boolean integer specifying whether or not velocities include the Hubble flow.
Code lines: 18
Contained by: module merger_tree_read_importers
Modules used: numerical_constants_boolean

function: galacticusvelocitydispersionavailable
Description: Return true if halo velocity dispersions are available.
Code lines: 9
Contained by: module merger_tree_read_importers

function: galacticusvelocitymaximumavailable
Description: Return true if halo rotation curve velocity maxima are available.
Code lines: 9
Contained by: module merger_tree_read_importers

interface: importerunitconvert
Description: Unit convertors for GALACTICUS format tree importer.
Code lines: 5
Contained by: module merger_tree_read_importers

function: importerunitconvertID
Description: Convert a set of values for GALACTICUS internal units.
Code lines: 24
Contained by: module merger_tree_read_importers
Modules used: cosmology_functions cosmology_parameters galacticus_error
function: importerunitconvert2d
Description: Convert a set of values for GALACTICUS internal units.
Code lines: 25
Contained by: module merger_tree_read_importers
Modules used:

function: importerunitconvertscalar
Description: Convert a set of values for GALACTICUS internal units.
Code lines: 21
Contained by: module merger_tree_read_importers
Modules used:

type: importerunits
Code lines: 24
Contained by: module merger_tree_read_importers

function: importerunitsexponentiate
Description: Exponentiate importerUnits objects.
Code lines: 12
Contained by: module merger_tree_read_importers

function: importerunitsmultiply
Description: Multiply to importerUnits objects.
Code lines: 12
Contained by: module merger_tree_read_importers

interface: mergertreeimporter
Code lines: 3
Contained by: module merger_tree_read_importers

function: mergertreeimporterangularmomenta3davailablenull
Description: Return true if angular momenta (vectors) are available.
Code lines: 7
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

function: mergertreeimporterangularmomentaavailablenull
Description: Return true if angular momenta (magnitudes) are available.
Code lines: 7
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

function: mergertreeimporterangularmomentaincludesubhalosnull
Description: Returns a Boolean specifying whether halo angular momenta (or spins) include the contribution from their subhalos.
Code lines: 7
Contained by: module merger_tree_read_importers
19. Source Code Documentation

**Modules used:** galacticus_error

**type:** mergertreeimporterclass

- **Code lines:** 189
- **Contained by:** module merger_tree_read_importers

**subroutine:** mergertreeimporterclosenull

- **Description:** Closes the file.
- **Code lines:** 7
- **Contained by:** module merger_tree_read_importers
- **Modules used:** galacticus_error

**function:** mergertreeimporterconstructordefault

- **Description:** Return a pointer to the default mergerTreeImporter object.
- **Code lines:** 8
- **Contained by:** module merger_tree_read_importers

**function:** mergertreeimporterconstructornamed

- **Description:** Return a pointer to a newly created mergerTreeImporter object of the specified type.
- **Code lines:** 30
- **Contained by:** module merger_tree_read_importers
- **Modules used:** galacticus_error iso_varying_string

**function:** mergertreeimportercubelengthnull

- **Description:** Returns the length of the simulation cube.
- **Code lines:** 9
- **Contained by:** module merger_tree_read_importers
- **Modules used:** galacticus_error

**interface:** mergertreeimportergalacticus

- **Description:** Constructors for the GALACTICUS format merger tree importer class.
- **Code lines:** 3
- **Contained by:** module merger_tree_read_importers

**subroutine:** mergertreeimporterimportnull

- **Description:** Imports the $i^{th}$ tree.
- **Code lines:** 10
- **Contained by:** module merger_tree_read_importers
- **Modules used:** galacticus_error

**subroutine:** mergertreeimporterinitialize

- **Description:** Initialize the default mergerTreeImporter object.
- **Code lines:** 53
- **Contained by:** module merger_tree_read_importers
- **Modules used:** galacticus_error input_parameters iso_varying_string

**function:** mergertreeimporterisfinalizable

- **Description:** Return true if this object can be finalized.
19.1. Program units

function: mergertreeimportermassesincludesubhalosnull
Description: Returns a Boolean specifying whether halo masses include the contribution from their subhalos.
Code lines: 7
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

function: mergertreeimporternodecountnull
Description: Returns the number of nodes in the $i$th tree.
Code lines: 8
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

subroutine: mergertreeimporteropennull
Description: Opens the file.
Code lines: 8
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

function: mergertreeimporterparticlecountavailablenull
Description: Return true if particle counts are available.
Code lines: 7
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

function: mergertreeimporterpositionsareperiodicnull
Description: Returns a Boolean integer specifying whether positions are periodic.
Code lines: 7
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

function: mergertreeimporterpositionsavailablenull
Description: Return true if positions and/or velocities are available.
Code lines: 8
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

function: mergertreeimporterscaleradiiavailablenull
Description: Return true if scale radii are available.
Code lines: 7
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

function: mergertreeimporterspin3davailablenull
Description: Return true if spin (vectors) are available.
Code lines: 7
function: mergertreeimporterspinavailablenull
Description: Return true if spin (magnitudes) are available.
Code lines: 7
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

function: mergertreeimportersubhalotracecountnull
Description: Returns the length of a node’s subhalo trace.
Code lines: 8
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

subroutine: mergertreeimportersubhalotracenull
Description: Supplies epochs, positions, and velocities for traced subhalos.
Code lines: 10
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

interface: mergertreeimportersussing
Description: Constructors for the GALACTICUS format merger tree importer class.
Code lines: 3
Contained by: module merger_tree_read_importers

function: mergertreeimportertreecountnull
Description: Returns a count of the number of trees available.
Code lines: 7
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

function: mergertreeimportertreeindexnull
Description: Returns the index of the $i$th tree.
Code lines: 8
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

function: mergertreeimportertreesareselfcontainednull
Description: Returns a Boolean integer specifying whether trees are self-contained.
Code lines: 7
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

function: mergertreeimportertreesavesubhalosnull
Description: Returns a Boolean integer specifying whether or not the trees have subhalos.
Code lines: 7
Contained by: module merger_tree_read_importers
Modules used: galacticus_error
**function**: mergertreeimportertreeweightnull

*Description:* Returns the weight to assign to the $i^{th}$ tree.

*Code lines:* 8

*Contained by:* module `merger_tree_read_importers`

*Modules used:* `galacticus_error`

**function**: mergertreeimportervelocitiesincludehubbleflownull

*Description:* Returns a Boolean integer specifying whether velocities include the Hubble flow.

*Code lines:* 7

*Contained by:* module `merger_tree_read_importers`

*Modules used:* `galacticus_error`

**function**: mergertreeimportervelocitydispersionavailablenull

*Description:* Return true if halo velocity dispersions are available.

*Code lines:* 7

*Contained by:* module `merger_tree_read_importers`

*Modules used:* `galacticus_error`

**function**: mergertreeimportervelocitymaximumavailablenull

*Description:* Return true if rotation curve velocity maxima are available.

*Code lines:* 7

*Contained by:* module `merger_tree_read_importers`

*Modules used:* `galacticus_error`

**type**: nodedata

*Description:* Structure used to store raw data read from merger tree files.

*Code lines:* 8

*Contained by:* module `merger_tree_read_importers`

**type**: nodedatagalacticus

*Description:* Extension of the nodeData class for GALACTICUS format merger trees. Stores particle indices and counts for nodes.

*Code lines:* 3

*Contained by:* module `merger_tree_read_importers`

**type**: nodedatasussing

*Description:* Extension of the nodeData class for “Sussing Merger Trees” format merger trees [Srisawat et al., 2013].

*Code lines:* 2

*Contained by:* module `merger_tree_read_importers`

**function**: sussingangularmomenta3davailable

*Description:* Return true if angular momenta vectors are available.

*Code lines:* 7

*Contained by:* module `merger_tree_read_importers`

**function**: sussingangularmomentaavailable

*Description:* Return true if angular momenta are available.
function: sussingangularmomentaincludesubhalos
Description: Return a Boolean specifying whether or not the halo angular momenta include the contribution from subhalos.
Code lines: 8
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

subroutine: sussingclose
Description: Close a GALACTICUS format merger tree file.
Code lines: 6
Contained by: module merger_tree_read_importers

function: sussingcubelength
Description: Return the length of the simulation cube.
Code lines: 12
Contained by: module merger_tree_read_importers
Modules used: numerical_constants_astronomical numerical_constants_boolean

function: sussingdefaultconstructor
Description: Default constructor for the “Sussing Merger Trees” format [Srisawat et al., 2013] merger tree importer.
Code lines: 206
Contained by: module merger_tree_read_importers
Modules used: galacticus_error input_parameters

subroutine: sussingdestructor
Description: Destructor for the GALACTICUS format merger tree importer class.
Code lines: 9
Contained by: module merger_tree_read_importers
Modules used: memory_management

subroutine: sussingimport
Description: Import the $i^{th}$ merger tree.
Code lines: 26
Contained by: module merger_tree_read_importers
Modules used: memory_management

function: sussinginsubvolume
Description: Determine if a point lies within a subvolume of the simulation box (possibly with some buffering).
Code lines: 9
Contained by: module merger_tree_read_importers

function: sussinginsubvolumeld
Description: Determine if a point lies within the 1-D range of a subvolume of the simulation box (possibly with some buffering).
Code lines: 23
19.1. Program units

**function:** sussingmassesincludesubhalos

*Description:* Return a Boolean specifying whether or not the halo masses include the contribution from subhalos.

*Code lines:* 8

**Contained by:** module `merger_tree_read_importers`

**Modules used:** `numerical_constants_astronomical`

**function:** sussingnodecount

*Description:* Return a count of the number of nodes in the $i^{th}$ tree.

*Code lines:* 10

**Contained by:** module `merger_tree_read_importers`

**function:** sussingparticlecountavailable

*Description:* Return true if particle counts are available.

*Code lines:* 7

**Contained by:** module `merger_tree_read_importers`

**function:** sussingperiodicseparation

*Description:* Determine the separation between two points in a periodic cube.

*Code lines:* 13

**Contained by:** module `merger_tree_read_importers`

**function:** sussingpositionsareperiodic

*Description:* Return a Boolean integer specifying whether or not positions are periodic.

*Code lines:* 8

**Contained by:** module `merger_tree_read_importers`

**Modules used:** `numerical_constants_boolean`

**function:** sussingpositionsavailable

*Description:* Return true if positions and/or velocities are available.

*Code lines:* 8

**Contained by:** module `merger_tree_read_importers`

**function:** sussingreadhaloascii

*Description:* Read an ASCII halo definition.

*Code lines:* 76

**Contained by:** module `merger_tree_read_importers`

**Modules used:**

- `cosmology_functions`
- `cosmology_parameters`
- `file_utilities`
- `galacticus_display`
- `galacticus_error`
- `memory_management`
- `numerical_comparison`
- `numerical_constants_astronomical`
- `numerical_constants_boolean`
- `regular_expressions`
- `string_handling`
function: sussingscaleradiiavailable
Description: Return true if scale radii are available.
Code lines: 8
Contained by: module merger_tree_read_importers

function: sussingspin3davailable
Description: Return true if spins vectors are available.
Code lines: 8
Contained by: module merger_tree_read_importers

function: sussingspinavailable
Description: Return true if spins are available.
Code lines: 8
Contained by: module merger_tree_read_importers

subroutine: sussingsubhalotrace
Description: Returns a trace of subhalo position/velocity.
Code lines: 11
Contained by: module merger_tree_read_importers
Modules used: galacticus_error

function: sussingsubhalotracecount
Description: Returns the length of a subhalo trace.
Code lines: 10
Contained by: module merger_tree_read_importers

function: sussingtreecount
Description: Return a count of the number of trees available.
Code lines: 8
Contained by: module merger_tree_read_importers

function: sussingtreeindex
Description: Return the index of the i-th tree.
Code lines: 9
Contained by: module merger_tree_read_importers

subroutine: sussingtreeindicesread
Description: Read the tree indices.
Code lines: 888
Contained by: module merger_tree_read_importers
Modules used: array_utilities arrays_search
                  file_utilities galacticus_display
                  galacticus_error iso_c_binding
                  kind_numbers memory_management
                  numerical_constants_astronomical numerical_constants_prefixes
function: sussingtreesareselfcontained
Description: Return a Boolean integer specifying whether or not the trees are self-contained.
Code lines: 8
Contained by: module merger_tree_read_importers
Modules used: numerical_constants_boolean

function: sussingtreeshavesubhalos
Description: Return a Boolean integer specifying whether or not the trees have subhalos.
Code lines: 8
Contained by: module merger_tree_read_importers
Modules used: numerical_constants_boolean

function: sussingtreeweight
Description: Return the weight to assign to trees.
Code lines: 15
Contained by: module merger_tree_read_importers
Modules used: cosmology_functions galacticus_error numerical_constants_astronomical numerical_constants_boolean

function: sussingvalueisbad
Description: Determine if a value in a “Sussing” merger tree file is bad
Code lines: 13
Contained by: module merger_tree_read_importers

function: sussingvelocitiesincludehubbleflow
Description: Return a Boolean integer specifying whether or not velocities include the Hubble flow.
Code lines: 8
Contained by: module merger_tree_read_importers
Modules used: numerical_constants_boolean

function: sussingvelocitydispersionavailable
Description: Return true if halo velocity dispersions are available.
Code lines: 7
Contained by: module merger_tree_read_importers

function: sussingvelocitymaximumavailable
Description: Return true if halo rotation curve velocity maxima are available.
Code lines: 7
Contained by: module merger_tree_read_importers

file: merger_trees.construct.read.importer.SussingMergerTrees.F90
Description: An implementation of the merger tree importer class for GALACTICUS format merger tree files.
Code lines: 1791
Modules used: io_hdf5 iso_varying_string stateful_types
interface: mergertreeimportersussing
Description: Constructors for the GALACTICUS format merger tree importer class.
Code lines: 3
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

type: nodedata_sussing
Description: Extension of the nodeData class for “Sussing Merger Trees” format merger trees [Srisawat et al., 2013].
Code lines: 2
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

function: sussingangularmomenta3davailable
Description: Return true if angular momenta vectors are available.
Code lines: 7
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

function: sussingangularmomentaavailable
Description: Return true if angular momenta are available.
Code lines: 7
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

function: sussingangularmomentaincludesubhalos
Description: Return a Boolean specifying whether or not the halo angular momenta include the contribution from subhalos.
Code lines: 8
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: galacticus_error

subroutine: sussingclose
Description: Close a GALACTICUS format merger tree file.
Code lines: 6
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

function: sussingcubelength
Description: Return the length of the simulation cube.
Code lines: 12
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: numerical_constants_astronomical numerical_constants_boolean

function: sussingdefaultconstructor
Description: Default constructor for the “Sussing Merger Trees” format [Srisawat et al., 2013] merger tree importer.
Code lines: 206
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: galacticus_error input_parameters

subroutine: sussingdestructor
Description: Destructor for the GALACTICUS format merger tree importer class.
19.1. Program units

Code lines: 9
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: memory_management

subroutine: sussingimport
Description: Import the $i^{th}$ merger tree.
Code lines: 26
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: memory_management

function: sussinginsubvolume
Description: Determine if a point lies within a subvolume of the simulation box (possibly with some buffering).
Code lines: 9
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

function: sussinginsubvolume1d
Description: Determine if a point lies within the 1-D range of a subvolume of the simulation box (possibly with some buffering).
Code lines: 23
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: numerical_constants_astronomical

function: sussingmassesincludesubhalos
Description: Return a Boolean specifying whether or not the halo masses include the contribution from subhalos.
Code lines: 8
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: galacticus_error

function: sussingnodecount
Description: Return a count of the number of nodes in the $i^{th}$ tree.
Code lines: 10
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

subroutine: sussingopen
Description: Validate a GALACTICUS format merger tree file.
Code lines: 123
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: cosmology_functions cosmology_parameters
file_utilities galacticus_display
galacticus_error memory_management
numerical_comparison numerical_constants_astronomical
regular_expressions string_handling

function: sussingparticlecountavailable
Description: Return true if particle counts are available.
Code lines: 7
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
function: sussingperiodicseparation
Description: Determine the separation between two points in a periodic cube.
Code lines: 13
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

function: sussingpositionsareperiodic
Description: Return a Boolean integer specifying whether or not positions are periodic.
Code lines: 8
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: numerical_constants_boolean

function: sussingpositionsavailable
Description: Return true if positions and/or velocities are available.
Code lines: 8
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

subroutine: sussingreadhaloascii
Description: Read an ASCII halo definition.
Code lines: 76
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: galacticus_error

function: sussingscaleradiiavailable
Description: Return true if scale radii are available.
Code lines: 8
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

function: sussingspin3davailable
Description: Return true if spins vectors are available.
Code lines: 8
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

function: sussingspinavailable
Description: Return true if spins are available.
Code lines: 8
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

subroutine: sussingsubhalotrace
Description: Returns a trace of subhalo position/velocity.
Code lines: 11
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: galacticus_error

function: sussingsubhalotracecount
Description: Returns the length of a subhalo trace.
Code lines: 10
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
function: sussingtreecount
Description: Return a count of the number of trees available.
Code lines: 8
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

function: sussingtreeindex
Description: Return the index of the $i^{th}$ tree.
Code lines: 9
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

subroutine: sussingtreeindicesread
Description: Read the tree indices.
Code lines: 888
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: array_utilities arrays_search
file_utilities galacticus_display
galacticus_error iso_c_binding
kind_numbers memory_management
numerical_constants_astronomical numerical_constants_prefixes
sort string_handling

function: sussingtreesareselfcontained
Description: Return a Boolean integer specifying whether or not the trees are self-contained.
Code lines: 8
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: numerical_constants_boolean

function: sussingtreeshavesubhalos
Description: Return a Boolean integer specifying whether or not the trees have subhalos.
Code lines: 8
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: numerical_constants_boolean

function: sussingtreeweight
Description: Return the weight to assign to trees.
Code lines: 15
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: cosmology_functions galacticus_error
numerical_constants_astronomical numerical_constants_boolean

function: sussingvalueisbad
Description: Determine if a value in a “Sussing” merger tree file is bad
Code lines: 13
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

function: sussingvelocitiesincludehubbleflow
Description: Return a Boolean integer specifying whether or not velocities include the Hubble flow.
Code lines: 8
19. Source Code Documentation

Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90
Modules used: numerical_constants_boolean

function: sussingvelocitydispersionavailable
Description: Return true if halo velocity dispersions are available.
Code lines: 7
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

function: sussingvelocitymaximumavailable
Description: Return true if halo rotation curve velocity maxima are available.
Code lines: 7
Contained by: file merger_trees.construct.read.importer.SussingMergerTrees.F90

file: merger_trees.construct.read.importer.galacticus.F90
Description: An implementation of the merger tree importer class for GALACTICUS format merger tree files.
Code lines: 1031
Modules used: io_hdf5 stateful_types

function: galacticusangularmomenta3davailable
Description: Return true if angular momenta vectors are available.
Code lines: 7
Contained by: file merger_trees.construct.read.importer.galacticus.F90

function: galacticusangularmomentaavailable
Description: Return true if angular momenta are available.
Code lines: 7
Contained by: file merger_trees.construct.read.importer.galacticus.F90

function: galacticusangularmomentaincludesubhalos
Description: Return a Boolean specifying whether or not the halo momenta include the contribution from subhalos.
Code lines: 24
Contained by: file merger_trees.construct.read.importer.galacticus.F90
Modules used: galacticus_error

subroutine: galacticusclose
Description: Validate a GALACTICUS format merger tree file.
Code lines: 11
Contained by: file merger_trees.construct.read.importer.galacticus.F90

function: galacticuscubelength
Description: Return the length of the simulation cube.
Code lines: 37
Contained by: file merger_trees.construct.read.importer.galacticus.F90
Modules used: galacticus_error numerical_constants_astronomical numerical_constants_boolean

function: galacticusdefaultconstructor
19.1. Program units

**Description:** Default constructor for the GALACTICUS format merger tree importer.

**Code lines:** 46

**Contained by:** file `merger_trees.construct.read.importer.galacticus.F90`

**Modules used:** `input_parameters`

**subroutine:** galacticusdestructor

**Description:** Destructor for the GALACTICUS format merger tree importer class.

**Code lines:** 7

**Contained by:** file `merger_trees.construct.read.importer.galacticus.F90`

**subroutine:** galacticusimport

**Description:** Import the $i$th merger tree.

**Code lines:** 190

**Contained by:** file `merger_trees.construct.read.importer.galacticus.F90`

**Modules used:** `cosmology_functions` `galacticus_display`
`galacticus_error` `hdf5`
`memory_management` `numerical_constants_astronomical`
`numerical_constants_prefixes` `vectors`

**function:** galacticusmassesincludesubhalos

**Description:** Return a Boolean specifying whether or not the halo masses include the contribution from subhalos.

**Code lines:** 20

**Contained by:** file `merger_trees.construct.read.importer.galacticus.F90`

**Modules used:** `galacticus_error`

**function:** galacticusnodecount

**Description:** Return a count of the number of nodes in the $i$th tree.

**Code lines:** 10

**Contained by:** file `merger_trees.construct.read.importer.galacticus.F90`

**subroutine:** galacticusopen

**Description:** Validate a GALACTICUS format merger tree file.

**Code lines:** 180

**Contained by:** file `merger_trees.construct.read.importer.galacticus.F90`

**Modules used:** `cosmology_parameters` `galacticus_display`
`galacticus_error` `numerical_comparison`
`power_spectra`

**function:** galacticusparticlecountavailable

**Description:** Return true if particle counts are available.

**Code lines:** 9

**Contained by:** file `merger_trees.construct.read.importer.galacticus.F90`

**function:** galacticuspositionsareperiodic

**Description:** Return a Boolean integer specifying whether or not positions are periodic.

**Code lines:** 18

**Contained by:** file `merger_trees.construct.read.importer.galacticus.F90`

**Modules used:** `numerical_constants_boolean`
function: galacticuspositionsavailable
Description: Return true if positions and/or velocities are available.
Code lines: 12
Contained by: file merger_trees.construct.read.importer.galacticus.F90

function: galacticusscaleradiiavailable
Description: Return true if scale radii are available.
Code lines: 9
Contained by: file merger_trees.construct.read.importer.galacticus.F90

function: galacticusspin3davailable
Description: Return true if spins vectors are available.
Code lines: 7
Contained by: file merger_trees.construct.read.importer.galacticus.F90

function: galacticusspinavailable
Description: Return true if spins are available.
Code lines: 7
Contained by: file merger_trees.construct.read.importer.galacticus.F90

subroutine: galacticussubhalotrace
Description: Returns a trace of subhalo position/velocity.
Code lines: 42
Contained by: file merger_trees.construct.read.importer.galacticus.F90
Modules used: cosmology_functions galacticus_error
               numerical_constants_astronomical

function: galacticussubhalotracecount
Description: Returns the length of a subhalo trace.
Code lines: 15
Contained by: file merger_trees.construct.read.importer.galacticus.F90
Modules used: galacticus_error

function: galacticustreecount
Description: Return a count of the number of trees available.
Code lines: 9
Contained by: file merger_trees.construct.read.importer.galacticus.F90

function: galacticustreeindex
Description: Return the index of the $i^{th}$ tree.
Code lines: 9
Contained by: file merger_trees.construct.read.importer.galacticus.F90

subroutine: galacticustreeindicesread
Description: Read the tree indices.
Code lines: 104
Contained by: file merger_trees.construct.read.importer.galacticus.F90
19.1. Program units

Module used: \texttt{cosmology\_functions} \hspace{1cm} \texttt{galacticus\_error}
\texttt{halo\_mass\_function} \hspace{1cm} \texttt{hdf5}
\texttt{numerical\_constants\_astronomical} \hspace{1cm} \texttt{sort}

\textbf{function: galacticustreesareselfcontained}
\textit{Description:} Return a Boolean integer specifying whether or not the trees are self-contained.
\textit{Code lines:} 18
\textit{Contained by:} file \texttt{merger\_trees.construct.read.importer.galacticus.F90}
\textit{Modules used:} \texttt{numerical\_constants\_boolean}

\textbf{function: galacticustreehassubhalos}
\textit{Description:} Return a Boolean integer specifying whether or not the trees have subhalos.
\textit{Code lines:} 18
\textit{Contained by:} file \texttt{merger\_trees.construct.read.importer.galacticus.F90}
\textit{Modules used:} \texttt{numerical\_constants\_boolean}

\textbf{function: galacticustreeweight}
\textit{Description:} Return the weight to assign to trees.
\textit{Code lines:} 33
\textit{Contained by:} file \texttt{merger\_trees.construct.read.importer.galacticus.F90}
\textit{Modules used:} \texttt{cosmology\_functions} \hspace{1cm} \texttt{galacticus\_error}
\texttt{numerical\_constants\_astronomical} \hspace{1cm} \texttt{numerical\_constants\_boolean}

\textbf{function: galacticusvelocitiesincludehubbleflow}
\textit{Description:} Return a Boolean integer specifying whether or not velocities include the Hubble flow.
\textit{Code lines:} 18
\textit{Contained by:} file \texttt{merger\_trees.construct.read.importer.galacticus.F90}
\textit{Modules used:} \texttt{numerical\_constants\_boolean}

\textbf{function: galacticusvelocitydispersionavailable}
\textit{Description:} Return true if halo velocity dispersions are available.
\textit{Code lines:} 9
\textit{Contained by:} file \texttt{merger\_trees.construct.read.importer.galacticus.F90}

\textbf{function: galacticusvelocitymaximumavailable}
\textit{Description:} Return true if halo rotation curve velocity maxima are available.
\textit{Code lines:} 9
\textit{Contained by:} file \texttt{merger\_trees.construct.read.importer.galacticus.F90}

\textbf{interface: mergertreeimportergalacticus}
\textit{Description:} Constructors for the GALACTICUS format merger tree importer class.
\textit{Code lines:} 3
\textit{Contained by:} file \texttt{merger\_trees.construct.read.importer.galacticus.F90}

\textbf{type: nodedatagalacticus}
\textit{Description:} Extension of the \texttt{nodeData} class for GALACTICUS format merger trees. Stores particle indices and counts for nodes.
\textit{Code lines:} 3
\textit{Contained by:} file \texttt{merger\_trees.construct.read.importer.galacticus.F90}
19. Source Code Documentation

file: merger_trees.construct.read.state.F90
Description: Contains a module which stores internal state for the merger tree reading module.
Code lines: 57

module: merger_tree_read_state
Description: Stores internal state for the merger tree reading module.
Code lines: 37
Contained by: file merger_trees.construct.read.state.F90
Used by: subroutine galacticus_state_retrieve subroutine galacticus_state_store

subroutine: merger_tree_read_state_retrieve
Description: Write the stored snapshot of the random number state to file.
Code lines: 9
Contained by: module merger_tree_read_state
Modules used: fgsl

subroutine: merger_tree_read_state_store
Description: Write the stored snapshot of the random number state to file.
Code lines: 9
Contained by: module merger_tree_read_state
Modules used: fgsl

file: merger_trees.construct.smooth_accretion.F90
Description: Contains a module which implements building of simple merger trees with smooth mass accretion histories and no branches using the fitting function of Wechsler et al. [2002].
Code lines: 170

module: merger_tree_smooth_accretion
Description: Implements building of simple merger trees with smooth mass accretion histories and no branches using the fitting function of Wechsler et al. [2002].
Code lines: 149
Contained by: file merger_trees.construct.smooth_accretion.F90
Modules used: iso_varying_string
Used by: function merger_tree_create

subroutine: merger_tree_smooth_accretion_do
Description: Build a merger tree with a smooth mass accretion history using the fitting function of Wechsler et al. [2002].
Code lines: 68
Contained by: module merger_tree_smooth_accretion
Modules used: cosmology_functions dark_matter_halo_mass_accretion_histories galacticus_nodes kind_numbers

subroutine: merger_tree_smooth_accretion_initialize
Description: Initializes the smooth accretion merger tree module.
Code lines: 58
Contained by: module merger_tree_smooth_accretion
19.1. Program units

**Modules used:**  input_parameters

**file:** merger_trees.construct.state_restore.F90

*Description:* Contains a module which implements storing and restoring of the complete internal state of a merger tree.

*Code lines:* 263

**module:** merger_trees_state_store

*Description:* Implements storing and restoring of the complete internal state of a merger tree. Useful primarily for debugging purposes to begin running a tree from just prior to the point of failure.

*Code lines:* 243

*Contained by:* file merger_trees.construct.state_restore.F90

*Modules used:* iso_varying_string kind_numbers

*Used by:* function merger_tree_create

**subroutine:** merger_tree_state_restore

*Description:* Restores the state of a merger tree from file.

*Code lines:* 76

*Contained by:* module merger_trees_state_store

*Modules used:* galacticus_error galacticus_nodes galacticus_state string_handling

**subroutine:** merger_tree_state_store

*Description:* Store the complete internal state of a merger tree to file.

*Code lines:* 63

*Contained by:* module merger_trees_state_store

*Modules used:* galacticus_nodes galacticus_state

**subroutine:** merger_tree_state_store_initialize

*Description:* Initialize the “state restore” method for constructing merger trees.

*Code lines:* 27

*Contained by:* module merger_trees_state_store

*Modules used:* input_parameters

**subroutine:** merger_tree_state_walk_tree

*Description:* Walk a merger tree for the purposes of storing the full state to file. Includes walking of formation nodes.

*Code lines:* 17

*Contained by:* module merger_trees_state_store

*Modules used:* galacticus_nodes

**function:** node_array_position

*Description:* Returns the position of a node in the output list given its index.

*Code lines:* 15

*Contained by:* module merger_trees_state_store

**function:** pointed_at_node

*Description:* Return a pointer to a node, given its position in the array of nodes. Return a null pointer if the array index is −1.
19. Source Code Documentation

Code lines: 13  
Contained by: module merger_trees_state_store  
Modules used: galacticus_nodes

file: merger_trees.dump_evolution.F90  
Description: Contains a module which dumps the evolution of merger trees to XML.  
Code lines: 111

module: merger_trees_dump_evolution  
Description: Dumps the structure of entire merger trees.  
Code lines: 91  
Contained by: file merger_trees.dump_evolution.F90  
Modules used: iso_varying_string  
Used by: subroutine galacticus_output_close_file subroutine tree_node_evolve

subroutine: merger_tree_dump_evolution  
Description: Trim histories attached to the disk.  
Code lines: 51  
Contained by: module merger_trees_dump_evolution  
Modules used: galacticus_nodes input_parameters

subroutine: merger_tree_dump_evolution_close  
Description: Close the merger tree evolution dump file.  
Code lines: 9  
Contained by: module merger_trees_dump_evolution

file: merger_trees.dump_structure.F90  
Description: Contains a module which dumps the structure of entire merger trees.  
Code lines: 129

module: merger_tree_dump_structure  
Description: Dumps the structure of entire merger trees.  
Code lines: 109  
Contained by: file merger_trees.dump_structure.F90  
Modules used: iso_varying_string  
Used by: function galacticus_task_evolve_tree

subroutine: merger_tree_structure_dump  
Description: Output the structure of thisTree.  
Code lines: 83  
Contained by: module merger_tree_dump_structure  
Modules used: galacticus_nodes input_parameters merger_trees_dump

file: merger_trees.evolve.F90  
Description: Contains a module which implements evolution of merger trees.  
Code lines: 824

module: merger_trees_evolve
19.1. Program units

**Description:** Implements evolution of merger trees.
**Code lines:** 804
**Contained by:** file `merger_trees.evolve.F90`
**Modules used:** `galacticus_nodes` `iso_varying_string` `kind_numbers`
**Used by:** function `galacticus_task_evolve_tree`

**subroutine:** `deadlock_add_node`
**Description:** Add a node to the deadlocked nodes list.
**Code lines:** 26
**Contained by:** module `merger_trees_evolve`

**subroutine:** `deadlock_tree_output`
**Description:** Output the deadlocked nodes in dot format.
**Code lines:** 93
**Contained by:** module `merger_trees_evolve`

**type:** `deadlocklist`
**Code lines:** 5
**Contained by:** module `merger_trees_evolve`

**function:** `evolve_to_time`
**Description:** Determine the time to which `thisNode` should be evolved.
**Code lines:** 227
**Contained by:** module `merger_trees_evolve`
**Modules used:** `cosmology_functions` `evolve_to_time_reports` `galacticus_display` `galacticus_error` `input_parameters` `merger_tree_timesteps`
**subroutine:** `merger_tree_evolve_to`
**Description:** Evolves all properties of a merger tree to the specified time.
**Code lines:** 326
**Contained by:** module `merger_trees_evolve`
**Modules used:** `galacticus_display` `galacticus_error` `input_parameters` `merger_trees_dump` `merger_trees_evolve_node` `merger_trees_timesteps_template` `node_component_hot_halo_cold_mode` `node_component_hot_halo_standard` `string_handling`

**subroutine:** `perform_node_events`
**Description:** Perform any events associated with `thisNode`.
**Code lines:** 46
**Contained by:** module `merger_trees_evolve`
subroutine: perform_tree_events
Description: Perform any events associated with thisNode.
Code lines: 41
Contained by: module merger_trees_evolve

file: merger_trees.evolve.deadlock_options.F90
Description: Contains a module which provides an enumeration for tree deadlock statuses.
Code lines: 23

module: merger_trees_evolve.deadlock_status
Description: Provides an enumeration for tree deadlock statuses.
Code lines: 3
Contained by: file merger_trees.evolve.deadlock_options.F90
Used by: function node_branch_jump function node_subhalo_promotion subroutine merger_tree_evolve_to subroutine satellite_merger_process

file: merger_trees.evolve.node.F90
Description: Contains a module which implements evolution of a single node in a merger tree.
Code lines: 861

module: merger_trees_evolve.node
Description: Implements evolution of a single node in a merger tree.
Code lines: 841
Contained by: file merger_trees.evolve.node.F90
Modules used: fodeiv2 galacticus_nodes iso_varying_string kind_numbers
Used by: function node_subhalo_promotion function evolve_to_time subroutine merger_tree_evolve_to

subroutine: events_node_merger
Description: Handles instances where thisNode is about to merge with its parent node.
Code lines: 108
Contained by: module merger_trees_evolve_node
Modules used: events_node_mergers_slh galacticus_error galacticus_display node_component_basic_standard node_component_host_history_standard node_component_hot_halo_cold_mode node_component_hot_halo_standard node_component_hot_halo_very_simple node_component_merging_statistics_recent node_component_merging_statistics_standard node_component_satellite_orbiting node_component_satellite_standard node_component_satellite_very_simple string_handling

subroutine: tree_node_compute_derivatives
Description: Call routines to set alls derivatives for thisNode.
Code lines: 112
19.1. Program units

**Contained by:** module `merger_trees_evolve_node`

**Modules used:**
- `galactic_structure_radii`
- `node_component_age_statistics_standard`
- `node_component_basic_standard`
- `node_component_black_hole_standard`
- `node_component_dark_matter_profile_scale`
- `node_component_dark_matter_profile_scale_preset`
- `node_component_disk_exponential`
- `node_component_formutation_times_cole2000`
- `node_component_hot_halo_outflow_tracking`
- `node_component_hot_halo very_simple`
- `node_component_inter_output_standard`
- `node_component_mass_flow_statistics_standard`
- `node_component Satellite_standard`
- `node_component_spin_preset`
- `odeiv2_solver`
- `iso_c_binding`
- `merger_trees_dump_evolution`
- `node_component_black_hole_simple`
- `node_component_dark_matter_profile_scale`
- `node_component_dark_matter_profile_scale_shape`
- `node_component_disk very_simple`
- `node_component_host_history_standard`
- `node_component_hot_halo_cold_mode`
- `node_component_hot_halo_outflow_tracking`
- `node_component_hot_halo_standard`
- `node_component_hot_halo_very_simple`
- `node_component_spheroid_standard`
- `node_component_spheroid_preset3d`

**subroutine:** `tree_node_evolve`

**Description:** Evolves `thisNode` to time `endTime`, or until evolution is interrupted.

**Code lines:** 229

**Contained by:** module `merger_trees_evolve_node`

**Modules used:**
- `galacticus_calculations_resets`
- `memory_management`
- `node_component_age_statistics_standard`
- `node_component_basic_standard`
- `node_component_black_hole_standard`
- `node_component_dark_matter_profile_scale`
- `node_component_dark_matter_profile_scale_preset`
- `node_component_disk_exponential`
- `node_component_formutation_times_cole2000`
- `node_component_hot_halo_outflow_tracking`
- `node_component_hot_halo very_simple`
- `node_component_mass_flow_statistics_standard`
- `node_component_satellite_standard`
- `node_component_host_history_standard`
- `node_component_hot_halo_cold_mode`
- `node_component_hot_halo_outflow_tracking`
- `node_component_hot_halo_standard`
- `node_component_hot_halo_very_simple`
- `node_component_spheroid_standard`
- `node_component_spheroid_preset3d`
- `odeiv2_solver`
- `iso_c_binding`
- `merger_trees_dump_evolution`
- `node_component_black_hole_simple`
- `node_component_dark_matter_profile_scale`
- `node_component_dark_matter_profile_scale_shape`
- `node_component_disk very_simple`
- `node_component_host_history_standard`
- `node_component_hot_halo_cold_mode`
- `node_component_hot_halo_outflow_tracking`
- `node_component_hot_halo_standard`
- `node_component_hot_halo_very_simple`
- `node_component_spheroid_standard`
- `node_component_spheroid_preset3d`

**subroutine:** `tree_node_evolve_error_analyzer`

**Description:** Profiles ODE solver step sizes and errors.

**Code lines:** 39

**Contained by:** module `merger_trees_evolve_node`
subroutine: tree_node_evolve_initialize
   Description: Initializes the tree evolving routines by reading in parameters
   Code lines: 81
   Contained by: module merger_trees_evolve_node
   Modules used: galacticus_error input_parameters

function: tree_node_is_accurate
   Description: Return true if a tree node property is within expected accuracy of a given value.
   Code lines: 12
   Contained by: module merger_trees_evolve_node
   Modules used: numerical_comparison

function: tree_node_odes
   Description: Function which evaluates the set of ODEs for the evolution of a specific node.
   Code lines: 49
   Contained by: module merger_trees_evolve_node
   Modules used: iso_c_binding ode_solver_error_codes

subroutine: tree_node_odes_error_handler
   Description: Handles errors in the ODE solver when evolving GALACTICUS nodes. Dumps the content of the node.
   Code lines: 12
   Contained by: module merger_trees_evolve_node
   Modules used: galacticus_display string_handling

subroutine: tree_node_promote
   Description: Transfer the properties of thisNode to its parent node, then destroy it.
   Code lines: 124
   Contained by: module merger_trees_evolve_node
   Modules used: galacticus_display node_component_basic_non_evolving
   node_component_basic_standard node_component_basic_standard_tracking
   node_component_dark_matter_profile_scale node_component_dark_matter_profile_scale_preset
   node_component_dark_matter_profile_scale_shape node_component_dark_matter_profile_scale_preset_shape
   node_component_host_history_standard node_component_hot_halo_cold_mode
   node_component_hot_halo_standard node_component_hot_halo_very_simple
   node_component_merging_statistics_recent node_component_merging_statistics_standard
   node_component_nbody_standard node_component_position_preset
   node_component_satellite_preset node_component_spin_preset
   node_component_spin_preset3d node_component_spin_random
   node_promotion_index_shifts string_handling

file: merger_trees.evolve.timesteps.F90
   Description: Contains a module which implements calculations of timesteps for merger tree evolution.
   Code lines: 81
module: merger_tree_timesteps
Description: Implements calculations of timesteps for merger tree evolution.
Code lines: 61
Contained by: file merger_trees.evolve.timesteps.F90
Used by: function evolve_to_time

function: time_step_get
Description: Computes a suitable timestep over which to evolve a node in a tree.
Code lines: 51
Contained by: module merger_tree_timesteps
Modules used: iso_varying_string merger_tree_timesteps_history
merger_tree_timesteps_record_evolution
merger_tree_timesteps_satellite
merger_tree_timesteps_simple
merger_trees_evolve_timesteps_template
node_component_dynamics_statistics_bars

file: merger_trees.evolve.timesteps.history.F90
Description: Contains a module which implements a time-stepping criterion for merger tree evolution which permits global histories to be stored.
Code lines: 384

module: merger_tree_timesteps_history
Description: Implements a simple time-stepping criterion for merger tree evolution.
Code lines: 363
Contained by: file merger_trees.evolve.timesteps.history.F90
Modules used: fgsl galacticus_nodes
Used by: subroutine galacticus_output_close_file function time_step_get

subroutine: merger_tree_history_store
Description: Store various properties in global arrays.
Code lines: 55
Contained by: module merger_tree_timesteps_history
Modules used: galactic_structure_enclosed_masses galactic_structure_options
iso_c_binding numerical_interpolation

subroutine: merger_tree_history_write
Description: Store the global history data to the GALACTICUS output file.
Code lines: 150
Contained by: module merger_tree_timesteps_history
Modules used: galacticus_hdf5 numerical_constants_astronomical

subroutine: merger_tree_timestep_history
Description: Determines the timestep to go to the next tabulation point for global history storage.
Code lines: 122
Contained by: module merger_tree_timesteps_history
file: merger_trees.evolve.timesteps.record_evolution.F90
Description: Contains a module which implements a time-stepping criterion for merger tree evolution which permits evolution of the main branch galaxy to be stored.
Code lines: 274

module: merger_tree_timesteps_record_evolution
Description: Implements a time-stepping criterion for merger tree evolution which permits evolution of the main branch galaxy to be stored.
Code lines: 253
Contained by: file merger_trees.evolve.timesteps.record_evolution.F90
Modules used: fgsl
Used by: subroutine galacticus_merger_tree_record_evolution_output, function time_step_get_output

subroutine: merger_tree_record_evolution_output
Description: Store Fourier-space halo profiles to the output file.
Code lines: 50
Contained by: module merger_tree_timesteps_record_evolution
Modules used: galacticus_hdf5, galacticus_nodes, galacticus_output_times, iso_c_binding, iso_varying_string, numerical_constants_astronomical, string_handling

subroutine: merger_tree_record_evolution_store
Description: Store properties of the main progenitor galaxy.
Code lines: 31
Contained by: module merger_tree_timesteps_record_evolution
Modules used: galactic_structure_enclosed_masses, galactic_structure_options, galactic_nodes, iso_c_binding, numerical_interpolation

subroutine: merger_tree_timestep_record_evolution
Description: Determines the timestep to go to the next tabulation point for galaxy evolution storage.
Code lines: 121
Contained by: module merger_tree_timesteps_record_evolution
Modules used: cosmology_functions, evolve_to_time_reports, input_parameters, iso_c_binding, iso_varying_string, memory_management, merger_trees_evolve_timesteps_template, numerical_interpolation, numerical_ranges
subroutine: reset_records
Description: Resets recorded datasets to zero.
Code lines: 8
Contained by: module merger_tree_timesteps_record_evolution

file: merger_trees.evolve.timesteps.report.F90
Description: Contains a module which reports on timestepping criteria.
Code lines: 50

module: evolve_to_time_reports
Description: Contains functions which report on timestepping criteria.
Code lines: 30
Contained by: file merger_trees.evolve.timesteps.report.F90
Used by: function evolve_to_time subroutine merger_tree_timestep_history subroutine merger_tree_timestep_- record_evolution subroutine merger_tree_timestep_- satellite subroutine merger_tree_timestep_simple subroutine node_component_dynamics_- statistics_bars_timestep

subroutine: evolve_to_time_report
Description: Display a report on evolution timestep criteria.
Code lines: 20
Contained by: module evolve_to_time_reports
Modules used: galacticus_display iso_varying_string kind_numbers string_handling

file: merger_trees.evolve.timesteps.satellite.F90
Description: Contains a module which implements a time-stepping criterion for merger tree evolution which stops evolution when a merger is about to happen.
Code lines: 231

module: merger_tree_timesteps_satellite
Code lines: 210
Contained by: file merger_trees.evolve.timesteps.satellite.F90
Used by: function time_step_get

subroutine: merger_tree_timestep_satellite
Description: Determines the timestep to go to the time at which the node merges.
Code lines: 107
Contained by: module merger_tree_timesteps_satellite
Modules used: evolve_to_time_reports input_parameters iso_varying_string merger_trees_evolve_timesteps_- template

subroutine: satellite_merger_process
Description: Process a satellite node which has undergone a merger with its host node.
Code lines: 80
Contained by: module merger_tree_timesteps_satellite
19. Source Code Documentation

Modules used:
- galacticus_display
- iso_varying_string
- node_component_age_statistics_standard
- node_component_black_hole_standard
- node_component_black_hole_simple
- node_component_disk_exponential
- node_component_disk_very_simple
- node_component_hot_halo_cold_mode
- node_component_hot_halo_standard
- node_component_hot_halo_very_simple
- node_component_inter_output_standard
- node_component_merging_statistics_standard
- node_component_spheroid_standard
- satellite_merging_mass_movements
- string_handling

file: merger_trees.evolve.timesteps.simple.F90
Description: Contains a module which implements a simple time-stepping criterion for merger tree evolution.
Code lines: 113

module: merger_tree_timesteps_simple
Description: Implements a simple time-stepping criterion for merger tree evolution.
Code lines: 93
Contained by: file merger_trees.evolve.timesteps.simple.F90
Used by: function time_step_get

subroutine: merger_tree_timestep_simple
Description: Determine a suitable timestep for thisNode using the simple method. This simply selects the smaller of timestepSimpleAbsolute and timestepSimpleRelativeH^{-1}(t).
Code lines: 74
Contained by: module merger_tree_timesteps_simple
Modules used:
- cosmology_functions
- evolve_to_time_reports
- galacticus_nodes
- input_parameters
- iso_varying_string

file: merger_trees.evolve.timesteps.template.F90
Description: Contains a module which defines the template for tasks performed at the end of timesteps.
Code lines: 35

module: merger_trees_evolve.timesteps_template
Description: Defines the template for tasks performed at the end of timesteps.
Code lines: 15
Contained by: file merger_trees.evolve.timesteps.template.F90
Modules used:
- galacticus_nodes
Used by:
- function evolve_to_time
- subroutine merger_tree_evolve_to
- function time_step_get
- subroutine merger_tree_timestep_history
- subroutine merger_tree_timestep_satellite
- subroutine merger_tree_timestep_record_evolution
- subroutine merger_tree_timestep_satellite

file: merger_trees.file_maker.Bolshoi.F90
Description: Contains a module which handles reading the Rockstar merger trees from the Bolshoi simulation.
Code lines: 132
module: merger_trees_bolshoi
Description: Handles reading the Rockstar merger trees from the Bolshoi simulation.
Code lines: 110
Contained by: file merger_trees.file_maker.Bolshoi.F90
Used by: program bolshoi_merger_tree_file_maker

subroutine: merger_trees_bolshoi_process
Description: Reads the Rockstar merger trees from the Bolshoi simulation.
Code lines: 100
Contained by: module merger_trees_bolshoi
Modules used:
  dates_and_times
  file_utilities
  input_parameters
  iso_varying_string
  merger_tree_data_structure
  numerical_constants_astronomical

file: merger_trees.file_maker.Millennium.F90
Description: Contains a module which handles reading of data from CSV files extracted from the Millennium Simulation database.
Code lines: 181

module: merger_trees_millennium
Description: Handles reading of data from CSV files extracted from the Millennium Simulation database.
Code lines: 161
Contained by: file merger_trees.file_maker.Millennium.F90
Used by: program millennium_merger_tree_file_maker

subroutine: merger_trees_millennium_process
Description: Read and process a CSV file of merger trees extracted from the Millennium Simulation database.
Code lines: 151
Contained by: module merger_trees_millennium
Modules used:
  dates_and_times
  file_utilities
  galacticus_error
  iso_varying_string
  merger_tree_data_structure
  numerical_constants_astronomical

file: merger_trees.file_maker.simple.F90
Description: Contains a module which handles reading of data from CSV files of simple merger trees.
Code lines: 177

module: merger_trees_simple
Description: Handles reading of data from CSV files of simple merger trees.
Code lines: 157
Contained by: file merger_trees.file_maker.simple.F90
Used by: program simple_merger_tree_file_maker

subroutine: merger_trees_simple_process
Description: Read and process a CSV file of simple merger trees.
Code lines: 147
19. Source Code Documentation

**Contained by:**  
module **merger_trees_simple**

**Modules used:**  
cosmology_parameters  
file_utilities  
dates_and_times  
input_parameters  
iso_varying_string  
merger_tree_data_structure  
numerical_constants_astronomical

**file:** merger_trees.initialize.F90  
*Description:* Contains a module which implements initialization of merger tree structures.  
*Code lines:* 114

**module:** merger_trees_initialize  
*Description:* Implements initialization of merger tree structures.  
*Code lines:* 94

**-contained by:**  
file: merger_trees.initialize.F90  
**Used by:** subroutine merger_tree_evolve_to

**subroutine:** merger_tree_initialize  
*Description:* Walk through all nodes of a tree and call any routines that requested to perform initialization tasks.  
*Code lines:* 84

**-contained by:**  
module: merger_trees_initialize  
**Modules used:**  
galacticus_nodes  
node_component_basic_standard  
node_component_basic_standard-tracking  
node_component_dark_matter_profile-scale  
node_component_dark_matter_profile-scale_preset  
node_component_dark_matter_profile-scale_shape  
node_component_formation_times-cole2000  
node_component_host_history_standard  
node_component_hot_halo_cold_mode  
node_component_hot_halo_standard  
node_component_hot_halo_very_simple  
node_component_indices_standard  
node_component_mass_flow_statistics-standard  
node_component_merging_statistics-recent  
node_component_merging_statistics-standard  
node_component_satellite_orbiting  
node_component_satellite_standard  
node_component_satellite_very_simple  
node_component_spin_preset  
node_component_spin_preset3d  
node_component_spin_random

**file:** merger_trees.mass_accretion_history.F90  
*Description:* Contains a module which outputs mass accretion histories of merger trees.  
*Code lines:* 155

**module:** merger_tree_mass_accretion_history  
*Description:* Outputs mass accretion histories of merger trees.  
*Code lines:* 135

**-contained by:**  
file: merger_trees.mass_accretion_history.F90  
**Used by:** subroutine galacticus_output_close_file  
**function:** galacticus_task_evolve_tree
subroutine: merger_tree_mass_accretion_history_close
Description: Close the mass accretion history group before closing the HDF5 file.
Code lines: 6
Contained by: module merger_tree_mass_accretion_history

subroutine: merger_tree_mass_accretion_history_output
Description: Output the mass accretion history of thisTree.
Code lines: 101
Contained by: module merger_tree_mass_accretion_history
Modules used: galacticus_error, galacticus_hdf5, galacticus_nodes, input_parameters, iso_c_binding, iso_varying_string, memory_management, numerical_constants_astronomical, string_handling

file: merger_trees.monotonic_mass_growth.F90
Description: Contains a module which enforces monotonic mass growth along merger tree branches.
Code lines: 110

module: merger_trees_monotonic_mass_growth
Description: Enforces monotonic mass growth along merger tree branches.
Code lines: 90
Contained by: file merger_trees.monotonic_mass_growth.F90
Used by: function galacticus_task_evolve_tree

subroutine: merger_tree_monotonic_mass_growth
Description: Enforce monotonic mass growth along branches of thisTree.
Code lines: 72
Contained by: module merger_trees_monotonic_mass_growth
Modules used: galacticus_nodes, input_parameters

file: merger_trees.output_structure.F90
Description: Contains a module which outputs the structure of entire merger trees.
Code lines: 295

module: merger_tree_output_structure
Description: Outputs the structure of entire merger trees.
Code lines: 275
Contained by: file merger_trees.output_structure.F90
Modules used: io_hdf5
Used by: subroutine galacticus_output_close_file, function galacticus_task_evolve_tree

subroutine: merger_tree_structure_output
Description: Output the structure of thisTree.
Code lines: 238
Contained by: module merger_tree_output_structure
Modules used: dark_matter_halo_scales, galacticus_hdf5
19. Source Code Documentation

<table>
<thead>
<tr>
<th>Subroutine/Module</th>
<th>Description</th>
<th>Code lines</th>
<th>Contained by</th>
<th>Used by</th>
</tr>
</thead>
<tbody>
<tr>
<td>merger_tree_output_structure</td>
<td>Close the merger tree structure group.</td>
<td>6</td>
<td>module merger_tree_output_structure</td>
<td></td>
</tr>
<tr>
<td>merger_trees.prune_branches.F90</td>
<td>Contains a module which prunes branches below a given mass threshold from merger trees.</td>
<td>138</td>
<td></td>
<td></td>
</tr>
<tr>
<td>merger_trees.prune_branches</td>
<td>Prunes branches below a given mass threshold from merger trees.</td>
<td>118</td>
<td>file merger_trees.prune_branches.F90</td>
<td>function galacticus_task_evolve_tree</td>
</tr>
<tr>
<td>merger_tree_prune_branches</td>
<td>Prune branches from thisTree.</td>
<td>96</td>
<td>module merger_trees_prune_branches</td>
<td></td>
</tr>
<tr>
<td>merger_trees.prune_hierarchy.F90</td>
<td>Contains a module which prunes hierarchy below a given depth in merger trees.</td>
<td>122</td>
<td></td>
<td></td>
</tr>
<tr>
<td>merger_trees.prune_hierarchy</td>
<td>Prunes hierarchy below a given depth in merger trees.</td>
<td>102</td>
<td>file merger_trees.prune_hierarchy.F90</td>
<td>function galacticus_task_evolve_tree</td>
</tr>
<tr>
<td>merger_tree_prune_hierarchy</td>
<td>Prune hierarchy from thisTree.</td>
<td>83</td>
<td>module merger_trees_prune_hierarchy</td>
<td></td>
</tr>
<tr>
<td>merger_trees.prune_non_essential.F90</td>
<td>Contains a module which prunes branches in a tree that do not contain an “essential” node.</td>
<td>142</td>
<td></td>
<td></td>
</tr>
<tr>
<td>merger_trees.prune_non_essential</td>
<td>Prunes branches in a tree that do not contain an “essential” node.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
19.1. Program units

**Code lines:** 122  
**Contained by:** file `merger_trees.prune_non_essential.F90`  
**Modules used:** `kind_numbers`  
**Used by:** function `galacticus_task_evolve_tree`

**subroutine:** `merger_tree_non_essential_branches`  
**Description:** Prune branches from `thisTree`.  
**Code lines:** 96  
**Contained by:** module `merger_trees_prune_non_essential`  
**Modules used:** `galacticus_nodes`  
**Used by:** function `galacticus_task_evolve_tree`

**file:** `merger_trees.regrid_times.F90`  
**Description:** Contains a module which prunes branches below a given mass threshold from merger trees.  
**Code lines:** 445

**module:** `merger_trees_regrid_times`  
**Description:** Forces a merger tree onto a specified time grid.  
**Code lines:** 425  
**Contained by:** file `merger_trees.regrid_times.F90`  
**Modules used:** `iso_varying_string`  
**Used by:** function `galacticus_task_evolve_tree`

**subroutine:** `merger_tree_regrid_time`  
**Description:** Regrid times of halos in `thisTree`.  
**Code lines:** 391  
**Contained by:** module `merger_trees_regrid_times`  
**Modules used:** `cosmology_functions`  
`critical_overdensity`  
`fgsl`  
`galacticus_error`  
`galacticus_nodes`  
`input_parameters`  
`iso_c_binding`  
`kind_numbers`  
`memory_management`  
`merger_trees_dump`  
`numerical_interpolation`  
`numerical_ranges`  
`sort`

**file:** `merger_trees.render.F90`  
**Description:** Contains a module which dumps information on merger tree structure useful for rendering 3D views of merger trees.  
**Code lines:** 134

**module:** `merger_trees_render`  
**Description:** Implements dumping of information on merger tree structure useful for rendering 3D views of merger trees.  
**Code lines:** 114  
**Contained by:** file `merger_trees.render.F90`  
**Modules used:** `kind_numbers`  

**subroutine:** `merger_trees_render_dump`  
**Description:** Dumps information on merger tree structure useful for rendering 3D views of merger trees.  
**Code lines:** 99
19. Source Code Documentation

Contained by: module merger_trees_render
Modules used: cosmology_functions dark_matter_halo_scales
               file_utilities galacticus_nodes
               io_hdf5 memory_management
               numerical_constants_astronomical

file: merger_trees.write.F90
Description: Contains a module which writes merger trees to file.
Code lines: 278

module: merger_trees_write
Description: Writes merger trees to file.
Code lines: 258
Contained by: file merger_trees.write.F90
Modules used: iso_varying_string
Used by: function galacticus_task_evolve_tree

subroutine: merger_tree_write
Description: Output the structure of thisTree.
Code lines: 229
Contained by: module merger_trees_write
Modules used: cosmology_functions cosmology_parameters
dates_and_times galacticus_nodes
               hdf5 input_parameters
               memory_management merger_tree_data_structure
               numerical_constants_astronomical numerical_interpolation
               power_spectra sort

file: numerical.ODE_solver.F90
Description: Contains a module which provides an interface to the GNU Scientific Library ODEIV differential equation solvers.
Code lines: 114

module: ode_solver
Description: Contains an interface to the GNU Scientific Library ODEIV differential equation solvers.
Code lines: 94
Contained by: file numerical.ODE_solver.F90
Modules used: ode_solver_error_codes
Used by: function matterdarkenergyconstructor matterdarkenergyexpansionfactorchange
         matterdarkenergyexpansionfactorchange
         matterdarkenergymakeexpansionfactor
         matterdarkenergymakeexpansionfactorchange
         matterdarkenergymakeexpansionfactorchange
         matterdarkenergymakeexpansionfactorchange
         dark_matter_halo_mass_accrimage_time_zhao2009
         linear_growth_factor_simple_tabulate
program test_ode_solver

subroutine: ode_solve
Description: Interface to the GNU Scientific Library ODEIV differential equation solvers.
Code lines: 68
Contained by: module ode_solver
Modules used: galacticus_error iso_c_binding

subroutine: ode_solver_free
Description: Free up workspace allocated to ODE solving.
Code lines: 13
Contained by: module ode_solver

file: numerical.ODE_solver.ODEIV2.F90
Description: Contains a module which provides an interface to the GNU Scientific Library ODEIV2 differential equation solvers.
Code lines: 148

module: odeiv2_solver
Description: Contains an interface to the GNU Scientific Library ODEIV2 differential equation solvers.
Code lines: 128
Contained by: file numerical.ODE_solver.ODEIV2.F90
Modules used: fodeiv2 ode_solver_error_codes
Used by: function intergalactic_medium_state internal_update subroutine tree_node_evolve
       function radiation_intergalactic background internal_update subroutine perturbation_dynamics_solver

subroutine: odeiv2_solve
Description: Interface to the GNU Scientific Library ODEIV2 differential equation solvers.
Code lines: 106
Contained by: module odeiv2_solver
Modules used: galacticus_error iso_c_binding
iso_varying_string string_handling

subroutine: odeiv2_solver_free
Description: Free up workspace allocated to ODE solving.
Code lines: 9
Contained by: module odeiv2_solver

file: numerical.ODE_solver.ODEIV2.wrapper.F90
Code lines: 513

module: fodeiv2
Code lines: 495
Contained by: file numerical.ODE_solver.ODEIV2.wrapper.F90
Modules used: fgs1 iso_c_binding
Used by: function intergalactic_medium_state internal_update subroutine tree_node_evolve
module merger_trees_evolve_node
module odeiv2_solver

subroutine perturbation_dynamics_solver

function radiation_intergalactic_background_internal_update

type: fodeiv2_control
Code lines: 2
Contained by: module fodeiv2

subroutine: fodeiv2_control_free
Code lines: 3
Contained by: module fodeiv2

function: fodeiv2_control_hadjust
Code lines: 7
Contained by: module fodeiv2

function: fodeiv2_control_init
Code lines: 5
Contained by: module fodeiv2

function: fodeiv2_control_name
Code lines: 7
Contained by: module fodeiv2

function: fodeiv2_control_scaled_new
Code lines: 6
Contained by: module fodeiv2

function: fodeiv2_control_standard_new
Code lines: 4
Contained by: module fodeiv2

function: fodeiv2_control_status
Code lines: 5
Contained by: module fodeiv2

function: fodeiv2_control_y_new
Code lines: 4
Contained by: module fodeiv2

function: fodeiv2_control_yp_new
Code lines: 4
Contained by: module fodeiv2

type: fodeiv2_driver
Code lines: 2
Contained by: module fodeiv2

function: fodeiv2_driver_alloc_scaled_new
Code lines: 9
19.1. Program units

**function**: fodeiv2_driver_alloc_standard_new

*Code lines:* 8

*Contained by:* module fodeiv2

**function**: fodeiv2_driver_alloc_y_new

*Code lines:* 8

*Contained by:* module fodeiv2

**function**: fodeiv2_driver_alloc_yp_new

*Code lines:* 8

*Contained by:* module fodeiv2

**function**: fodeiv2_driver_apply

*Code lines:* 11

*Contained by:* module fodeiv2

**subroutine**: fodeiv2_driver_free

*Code lines:* 3

*Contained by:* module fodeiv2

**function**: fodeiv2_driver_reset

*Code lines:* 4

*Contained by:* module fodeiv2

**function**: fodeiv2_driver_set_hmax

*Code lines:* 5

*Contained by:* module fodeiv2

**function**: fodeiv2_driver_set_hmin

*Code lines:* 5

*Contained by:* module fodeiv2

**function**: fodeiv2_driver_set_nmax

*Code lines:* 5

*Contained by:* module fodeiv2

**function**: fodeiv2_driver_status

*Code lines:* 5

*Contained by:* module fodeiv2

**type**: fodeiv2_evolve

*Code lines:* 2

*Contained by:* module fodeiv2

**function**: fodeiv2_evolve_alloc

*Code lines:* 4

*Contained by:* module fodeiv2
function: fodeiv2_evolve_apply
  Code lines: 10
  Contained by: module fodeiv2

subroutine: fodeiv2_evolve_free
  Code lines: 3
  Contained by: module fodeiv2

function: fodeiv2_evolve_reset
  Code lines: 4
  Contained by: module fodeiv2

function: fodeiv2_evolve_status
  Code lines: 5
  Contained by: module fodeiv2

type: fodeiv2_step
  Code lines: 2
  Contained by: module fodeiv2

function: fodeiv2_step_alloc
  Code lines: 12
  Contained by: module fodeiv2

function: fodeiv2_step_apply
  Code lines: 7
  Contained by: module fodeiv2

subroutine: fodeiv2_step_free
  Code lines: 3
  Contained by: module fodeiv2

function: fodeiv2_step_name
  Code lines: 7
  Contained by: module fodeiv2

function: fodeiv2_step_order
  Code lines: 4
  Contained by: module fodeiv2

function: fodeiv2_step_reset
  Code lines: 4
  Contained by: module fodeiv2

function: fodeiv2_step_status
  Code lines: 5
  Contained by: module fodeiv2
19.1. Program units

**type**: fodeiv2_step_type
  
  **Code lines**: 3
  
  **Contained by**: module fodeiv2

**type**: fodeiv2_system

**Code lines**: 3

**Contained by**: module fodeiv2

**subroutine**: fodeiv2_system_free

**Code lines**: 3

**Contained by**: module fodeiv2

**function**: fodeiv2_system_init

**Code lines**: 33

**Contained by**: module fodeiv2

**function**: fodeiv2_system_status

**Code lines**: 5

**Contained by**: module fodeiv2

**file**: numerical.ODE_solver.error_codes.F90

**Description**: Contains a module which defines internal error codes for the GALACTICUS ODE solver.

**Code lines**: 32

**module**: ode_solver_error_codes

**Description**: Defines internal error codes for the GALACTICUS ODE solver.

**Code lines**: 12

**Contained by**: file numerical.ODE_solver.error_codes.F90

**Modules used**: fgsl

**Used by**: function intergalactic_medium_state_internal_odes

**function**: tree_node_odes

**module**: ode_solver

**module**: odeiv2_solver

**function**: backgroundradiationodes

**file**: numerical.comparison.F90

**Description**: Contains a module which implements comparisons of values.

**Code lines**: 118

**module**: numerical_comparison

**Description**: Implements comparisons of values.

**Code lines**: 98

**Contained by**: file numerical.comparison.F90

**Used by**: subroutine chemical_state_cie_file_read

**subroutine**: cooling_function_cie_file_read

**function**: matterdarkenergyconstructor

**function**: matterlambdaconstructor

**subroutine**: galacticus_output_analysis_mass_dpndnt_sz_dstrbtins

**subroutine**: galacticus_output_analysis_mass_functions
function galacticus_output_time_index
subroutine intergalactic_medium_state_internal_initialize
subroutine merger_tree_conditional_mass_function
subroutine galacticusopen
subroutine sussingopen
function tree_node_is_accurate
subroutine mass_distribution_beta_profile_initialize
function mass_distribution_hernquist_density_radial_moment
function mass_distribution_sersic_density_radial_moment
subroutine satellite_merging_remnant_size_cole2000
subroutine critical_overdensity_kitayama_suto1996_initialize
function power_spectrum_nonlinear_cosmicemu
function bryannorman1998defaultconstructor
program test_comparision
subroutine assert_double_2d_array
subroutine assert_double_4d_array
subroutine assert_double_scalar
subroutine assert_real_1d_array

interface: values_agree
Code lines: 3
Contained by: module numerical_comparison

function: values_agree_double
Description: Returns true if value1 and value2 agree to within absTol in absolute terms, or relTol in relative terms.
Code lines: 24
Contained by: module numerical_comparison

function: values_agree_real
Description: Returns true if value1 and value2 agree to within absTol in absolute terms, or relTol in relative terms.
Code lines: 24
Contained by: module numerical_comparison

interface: values_differ
Code lines: 3
Contained by: module numerical_comparison

function: values_differ_double
Description: Returns true if value1 and value2 differ by more than absTol in absolute terms, or relTol in relative terms.
19.1. Program units

**Code lines:** 12  
**Contained by:** module **numerical_comparison**

**function:** **values_differ_real**  
**Description:** Returns true if **value1** and **value2** differ by more than **absTol** in absolute terms, or **relTol** in relative terms.  
**Code lines:** 12  
**Contained by:** module **numerical_comparison**

**file:** **numerical.constants.astronomical.F90**  
**Description:** Contains a module of useful astronomical constants.  
**Code lines:** 70

**module:** **numerical_constants_astronomical**  
**Description:** Contains various useful astronomical constants.  
**Code lines:** 50  
**Contained by:** file **numerical.constants.astronomical.F90**  
**Modules used:** **numerical_constants_atomic numerical_constants_units**  
**Used by:** program **mass_function_covariance**  
**function bondi_hoyle_lyttleton_accretion_rate**  
**function coldmodechemicalmasses**  
**funcation coldmodecoldmodefraction**  
**function simplechemicalmasses**  
**function accretion_disk_jet_power_shakura_sunyaev**  
**function filespectrum**  
**function black_hole_binary_separation_growth_rate_standard**  
**function chemicals_mass_to_density_conversion**  
**function electron_density_cie_file_interpolate**  
**function cooling_cold_mode_infall_output_names**  
**function cooling_function_cie_file_interpolate**  
**function cooling_radius_hot_halo_output_names**  
**function cooling_time_simple**  
**function cosmologyfunctionscomovingvolumeelement**  
**function matterlambdacomovingdistanceintegrand**  
**function hubbleconstantsimple**  
**function omegaradiationsimple**  
**function omegaradiationsimple**  
**function virialdensitycontrastdefinitiondynamicaltimescale**
19. Source Code Documentation

function virialdensitycontrastdefinition
function virialdensitycontrastdefinition_dynamical_time_scale
function einasto_free_fall_radius
function einasto_free_fall_radius_increasing_rate
function isothermal_free_fall_radius
function isothermal_free_fall_radius_increasing_rate
function nfw_free_fall_radius
function nfw_free_fall_radius_increasing_rate
subroutine bar_instability_timescale_eln
subroutine bar_instability_timescale_eln_tidal
subroutine meta_tree_timing_output

subroutine galacticus_output_analysis_mass_functions_output
subroutine galacticus_output_tree_density_contrast_names
subroutine galacticus_output_tree_lightcone_names
subroutine galacticus_output_tree_half_mass_names
subroutine galacticus_output_tree_mass_profile_names
subroutine galacticus_output_tree_satellite_extremum_names
subroutine galacticus_output_tree_satellite_status_names
subroutine galacticus_output_tree_virial_names
subroutine halo_mass_function_output

function intergalactic_medium_state_electron_scattering_integrand
function simple_electron_fraction
function simple_neutral_hydrogen_fraction
function recfast_constructor

function simple_neutral_helium_fraction
function simple_singly_ionized_helium_fraction
subroutine intergalactic_medium_state_internal_initialize
function intergalactic_medium_state_internal_odes
function simpleelectronfraction
function simplifyneutralhydrogenfraction

subroutine merger_tree_conditional_mass_function_output
function galacticuscubelength
subroutine galacticussubhalotrace
function galacticustreeweight
function sussinginsubvolumeld
subroutine sussingtreeindicesread
function sussingcubelength
subroutine sussingopen
function sussingtreeweight
subroutine galacticusimport
subroutine galacticussubhalotrace
function galacticustreeweight
subroutine merger_tree_history_write
subroutine merger_trees_bolshoi_process
subroutine merger_trees_millennium_process
subroutine merger_tree_mass_accretion_history_output
module abundances_structure
module galacticus_nodes
subroutine node_component_black_hole_simple_output_names
subroutine node_component_black_hole_standard_output_names
subroutine node_component_dark_matter_profile_scale_tree_output
subroutine node_component_dynamics_statistics_bars_output
subroutine node_component_hot_halo_cold_mode_outflow_return
subroutine node_component_hot_halo_cold_mode_rate_compute
subroutine node_component_hot_halo_standard_formation
subroutine node_component_hot_halo_standard_rate_compute
subroutine node_component_satellite_orbiting_scale_set
module tensors
subroutine radiation_igb_internal_flux
subroutine radiation_igb_internal_flux
function intergalactic_medium_state_internal_update
function simplifyneutralheliumfraction
function simplesinglyionizedheliumfraction
subroutine merger_tree_read_initialize
subroutine galacticusimport
subroutine galacticustreeindicesread
function sussingcubelength
subroutine sussingopen
function sussingtreeindicesread
function galacticuscubelength
subroutine galacticussubhalotrace
function galacticustreeweight
subroutine merger_tree_record_evolution_output
subroutine merger_trees_render_dump
subroutine merger_tree_write
module abundances_structure
module galacticus_nodes
subroutine node_component_black_hole_standard_mass_accretion_rate
subroutine node_component_black_hole_standard_rate_compute
subroutine node_component_disk_exponential_rate_compute
subroutine node_component_hot_halo_cold_mode_formation
subroutine node_component_hot_halo_cold_mode_rate_compute
subroutine node_component_hot_halo_standard_outflow_return
subroutine node_component_hot_halo_standard_rate_compute
subroutine node_component_satellite_orbiting_rate_compute
subroutine node_component_spheroid_standard_rate_compute
program optimal_sampling_smf
function backgroundradiationodes
function ram_pressure_stripping_mass_loss_rate_disk_simple
function ram_pressure_stripping_mass_loss_rate_spheroid_simple
function satellite_tidal_heating_rate_gnedin
function tidal_radius_heated_halo_solver
function imf_metal_yield_rate_noninstantaneous
subroutine star_formation_rate_surface_density_disks_br_initialize
function star_formation_timescale_spheroid_dynamical_time
subroutine supernovae_population_iii_hegerwoosley_initialize
function filter_luminosity_integral_ab
subroutine stellar_population_spectra_file_tabulation
function ideal_gas_sound_speed
function tidal_stripping_mass_loss_rate_disk_simple

function satellite_dynamical_friction_acceleration_chandrasekhar
function satellite_tidal_stripping_rate_zentner2005
function imf_energy_input_rate_noninstantaneous
function imf_recycling_rate_noninstantaneous
function star_formation_timescale_disk_dynamical_time
module stellar_feedback_standard
module stellar_astrophysics_winds_leitherer1992
function stellar_population_luminosity
subroutine transfer_function_camb_make
subroutine intergalactic_medium_state_internal_initialize
function intergalactic_medium_state_odes
module numerical_constants_astronomical
module chemical_structures
subroutine node_component_hot_halo_cold_mode_outflow_return
function background_radiation_odes
subroutine inoue2014apply
subroutine madau1995apply
subroutine meiksin2006apply
subroutine lycepspressapply
subroutine meiksin2006apply
program test_inoue2014

file: numerical.constants.atomic.F90
Description: Contains a module of useful atomic constants.
Code lines: 42

module: numerical.constants.atomic
Description: Contains various useful atomic constants.
Code lines: 22
Contained by: file numerical.constants.atomic.F90
Modules used: fgsl
numerical_constants_units
numerical_constants_physical
numerical_constants_units
NumericalConstants

Used by:
function coldmodecoldmodefraction
subroutine intergalactic_medium_state_internal_initialize
module numerical_constants_astronomical
subroutine node_component_hot_halo_cold_mode_outflow_return
function background_radiation_odes
subroutine inoue2014apply
subroutine madau1995apply
subroutine meiksin2006apply
subroutine lycepspressapply
subroutine meiksin2006apply
program test_inoue2014

file: numerical.constants.boolean.F90
Description: Contains a module of useful Boolean constants.
Code lines: 28
module: numerical_constants_boolean
Description: Contains various useful Boolean constants.
Code lines: 8
Contained by: file numerical.constants.boolean.F90
Used by: subroutine merger_tree_read_initialize
        function galacticuscubelength
        function galacticustreesareselfcontained
        function galacticustreeweight
        function sussingcube_length
        function sussingtreesareselfcontained
        function sussingtreeweight
        function galacticuscubelength
        function galacticustreesareselfcontained
        function galacticustreeweight
file: numerical.constants.math.F90
Description: Contains a module of useful mathematical constants.
Code lines: 43
module: numerical_constants_math
Description: Contains various useful mathematical constants.
Code lines: 23
Contained by: file numerical.constants.math.F90
Modules used: fgsl kind_numbers
Used by: program tests_excursion_sets
        function coldmodecoldmodefraction
        function black_hole_binary_recoil_-
            velocity_standard
        function black_hole_static_radius_spin
        function cooling_rate_white_frenk
        function densitycriticalsimple
        function einastoenergytablemake
        function einastorotationnormalization
        subroutine einastoenergytablemake
        function einastorotationnormalization
        subroutine einastoenergytablemake
        function einastorotationnormalization
        function isothermal_density
function nfdensityscalefree
function nfdensityscalefree
function isothermaldensity

function diemerkravtsov2014concentration
function prada2011inversesigmoidmin
function prada2011inversesigmoidmin
function dark_matter_profile_rotation_curve_gradient_task
function mean_density_contrast_root

function galacticus_output_trees_vlcty_dsprsn_lambdar_intgrnd1
function spherical_shellsolid_angle_in_cylcinder
subroutine geometry_survey_window_functions_caputi_2011_ukidss_uds
subroutine geometry_survey_window_functions_martin_2010_alfalfa
subroutine intergalactic_medium_state_internal_initialize
function intergalactic_medium_state_internal_update
module generalized_press_schechter_branching
module numerical_constants_physical
subroutine mass_distribution_beta_profile_initialize
function mass_distribution_beta_profile_potential
subroutine mass_distribution_hernquist_initialize
function mass_distribution_hernquist_mass_enc_by_sphere
function mass_distribution_sersic_density_radial_moment
function mass_distribution_sersic_mass_enc_by_sphere
subroutine sersic_profile_tabulate

function node_component_disk_exponential_density
function node_component_spheroid_standard_density
function node_component_spheroid_standard_rotation_curve_gradient

function nfwprofileenergy
function nfwprofileenergy
function
function diemerkravtsov2014concentration
function prada2011cmin
function prada2011cmin
function dark_matter_profile_scale
function galactic_structure_radius_initial_derivative_adiabatic_solver
function galacticus_output_trees_vlcty_dsprsn_lambdar_intgrnd1
function spherical_shellsolid_angle_in_cylcinder
subroutine geometry_survey_window_functions_caputi_2011_ukidss_uds
subroutine geometry_survey_window_functions_li_white_2009_sdss
subroutine geometry_survey_window_functions_martin_2010_alfalfa
subroutine intergalactic_medium_state_internal_odes
function intergalactic_medium_state_internal_odes
function gaussian_distribution

module modified_press_schechter_branching
module numerical_constants_units
subroutine mass_distribution_beta_profile_mass_enc_by_sphere
function mass_distribution_hernquist_density_radial_moment
function mass_distribution_hernquist_mass_enc_by_sphere
subroutine mass_distribution_nfw_initialize
subroutine mass_distribution_sersic_initialize
function sersic_abel_integrand

function node_component_disk_exponential_density
function node_component_spheroid_standard_density
function node_component_spheroid_standard_rotation_curve_bessel_factors
19.1. Program units

function node_component_disk_-exponential_rttn_crv_grdnt_bssl_fctrs
subroutine node_component_hot_halo_-cold_mode_outflow_return
subroutine node_component_satellite_-orbiting_create
subroutine power_spectrum_compute

function ram_pressure_stripping_mass_-loss_rate_spheroid_simple
function satellites_tidal_fields_-spherical_symmetry_get
function satellite_tidal_stripping_-rate_zentner2005
function star Formation_rate_surface_-density_disk_br
function star Formation_rate_surface_-density_disk_br
function star Formation_timescale_-disk_integrated_sd
function variance_integral

function erfapproximation

function excursion_sets_first_-crossing_rate_linear
subroutine initialize_cosmological_-mass_variance
function power_spectrum_dimensionless

subroutine power_spectrum_power_law_-tabulate
function power_spectrum_window_-function_top_hat
subroutine power_spectrum_window_-functions_th_kss_hybrid_initialize
subroutine transfer_function_-eisenstein_hu_make
subroutine transfer_function_file_read
function bryannorman1998densitycontrast
function bryannorman1998densitycontrast

function friendsoffriendsdensitycontrast
function kitayamasuto1996densitycontrast

subroutine node_component_dynamics_-statistics_bars_record
function node_component_hot_halo_-mode_rotation_curve_gradient_task
subroutine node_component_satellite_-orbiting_rate_compute
function radiation_intergalactic_-background_internal_update
function satellite_dynamical_friction_-acceleration_chandrasekhar
function satellite_tidal_heating_rate_-gnedin
function star formation_feedback_disk_-outflow_rate_creasey2012
function star formation_rate_surface_-density_disk_ks
subroutine mass function_covariance_-matrix
subroutine critical_overdensity_-kitayama_suto1996
function excursion_sets_first_-crossing_probability_linear
function halo_mass_function_sheth_-tormen_differential
function power_spectrum

function power_spectrum_nonlinear_-peacockdodds1996
subroutine power_spectrum_window_-functions_sharp_kspace_initialize
function power_spectrum_window_-function_th_kss_hybrid
subroutine transfer_function_bbks_make
function transfer_function_half_mode_-mass_eisenstein_hu
subroutine transfer_function_null_make
function bryannorman1998densitycontrastrateofchange
function bryannorman1998densitycontrastrateofchange
function friendsoffriendsdensitycontrastrateofchange
function kitayamasuto1996densitycontrastrateofchange
function kitayamasuto1996densitycontrast
function kitayamasuto1996densitycontrastrateofchange
function friendsoffriendsdensitycontrast
function friendsoffriendsdensitycontrastrateofchange
program tests_cosmic_age
program test_coordinate_systems
program tests_power_spectrum
program tests_sphericalCollapse__dark_energy_eds
program tests_sphericalCollapse__dark_energy_omega_two_thirds
program tests_sphericalCollapse__dark_energy_open
program tests_sphericalCollapse_open
function tidal_stripping_mass_loss_rate_disk_simple

file: numerical.constants.physical.F90
Description: Contains a module of useful physical constants.
Code lines: 67

module: numerical.constants.physical
Description: Contains various useful physical constants.
Code lines: 47
Contained by: file numerical.constants.physical.F90
Modules used: fgal numerical_constants_prefixes numerical_constants_math
Used by: function bondi_hoyle_lyttleton_accretion_radius function bondi_hoyle_lyttleton_accretion_rate
function coldmodecoldmodefraction function coldmodecoldmodefraction
function accretion_disk_adaf_tabulate function accretion_disk_jet_power_eddington
function filespectrum function filespectrum
function atomic_cross_section_compton function atomic_cross_section_ionization_photo_verner
function gaunt_factor_sutherland1998 function atomic_rate_ionization_collisional_verner
function black_hole_binary_initial_radius_volonteri_2003 function black_hole_binary_separation_growth_rate_standard
function black_hole_eddington_accretion_rate function black_hole_gravitational_radius
function black_hole_isco_specific_angular_momentum function black_hole_isco_specific_energy_node
subroutine chemical_hydrogen_rate_h2_gamma_to_2h subroutine chemical_hydrogen_rate_h2plus_electron
19.1. Program units

subroutine chemical_hydrogen_rate_h2_gamma_to_h2star_to_2h
subroutine chemical_hydrogen_rate_h2_hplus_to_h2plus_h
subroutine chemical_hydrogen_rate_h2plus_h2plus_to_h2plus_h
subroutine chemical_hydrogen_rate_h2_hplus_gamma_to_h_hplus
subroutine chemical_hydrogen_rate_h_hplus_hminus_to_2h_electron
function cross_section_h2_gamma_to_2h

function cross_section_h2plus_gamma_to_2hplus_electron
function cross_section_h2_h2plus_to_2hplus_electron
function h_hminus_to_2h_electron_rate_coefficient
function cooling_function_cmb_compton

function cooling_time_simple
function cooling_specific_angular_momentum_constant_rotation

function cosmologyfunctionscomovingvolumeelementtime
function matterlambdacomovingdistanceintegrand
function densitycriticalsimple
function densitycriticalsimple
function virialdensitycontrastdefinitionvirialtemperature
function virialdensitycontrastdefinitionvirialvelocity
function dark_matter_halo_angular_momentum
function einastocircularvelocitymaximum
function einastofreefallradius
function einastofreefallradiusincrease
function einastoradiusfromspecificangularmomentum
function einastocircularvelocitymaximum
function einastofreefallradius
function einastofreefallradiusincrease
function einastoradiusfromspecificangularmomentum
function nfwcircularvelocitymaximum
function dark_matter_profile_rotation_curve_gradient_task
function dark_matter_profile_rotation_curve_task
subroutine bar_instability_timescale_eln
subroutine solve_for_radius
function velocity DISPERSION integrand
function hot_halo_mass_distribution_rotation_curve_gradient_task
function hot_halo Rams_pressure_stripping_radius_solver
function intergalactic medium state electron scattering integrand
function kepler orbits energy
function kepler orbits velocity scale
function mass distribution hernquist potential
function node component disk exponential potential
function node component spheroid standard potential
function node component black hole simple potential
function node component black hole exponential rotation curve
function node component disk exponential rotation curve
function node component spheroid standard rotation curve
function node component black hole standard rotation curve
function node component disk exponential rotation curve
function node component black hole standard triple interaction
function node component black hole standard rotation curve
function node component disk exponential radius solver
function node component hot halo cold mode rotation curve task
function radiation integrate over cross section
function radiation igb internal flux
function radiation intergalactic background internal update
function ram pressure stripping mass loss rate disk simple
function ram pressure stripping mass loss rate spheroid simple
19.1. Program units

function satellite_dynamical_friction_acceleration_chandrasekhar
subroutine satellite_merging_remnant_size_coevinton2008
subroutine satellite_merging_remnant_progenitor_properties_standard
subroutine satellite_orbit_extremum_phase_space_coordinates
function satellite_tidal_heating_rate_gnedin
function tidal_radius_heated_halo_solver
subroutine star_formation_rate_surface_density_disks_br_initialize

program tests_kepler_orbits
function ideal_gas_sound_speed
function tidal_stripping_mass_loss_rate_disk_simple

file: numerical.constants.prefixes.F90
Description: Contains a module of useful numerical prefixes.
Code lines: 47

module: numerical_constants_prefixes
Description: Contains useful numerical prefixes.
Code lines: 27
Contained by: file numerical.constants-prefixes.F90
Used by: program xray_absorption_ism_wilms2000
function coldmodecoldmodefraction
function coldmodecoldmodefraction
subroutine cooling_function_temperature_slope_molecular_hydrogen_gp
function hubbleconstantsimple
module galacticus_output_analyses_mass_dpndnt_sz_dstrbtins
function intergalactic_medium_state_internal_odes
subroutine galacticusimport
subroutine sussingtreeindicesread
module numerical_constants_physical
subroutine store_unit_attributes_irate

function node_component_spheroid_standard_rotation_curve_gradient
subroutine node_component_hot_halo_standard_outflow_return

subroutine satellite_merging_remnant_size_cole2000
subroutine satellite_merging_remnant_progenitor_properties_cole2000
function satellite_orbit_convert_to_current_potential
function satellites_tidal_fields_spherical_symmetry_get
function satellite_tidal_stripping_rate_zentner2005
function starformation_rate_surface_density_disk_br
function starformation_rate_surface_density_disk_ks
function ideal_gas_jeans_length
function blackbody_emission
function tidal_stripping_mass_loss_rate_spheroid_simple
subroutine node_component_satellite_orbiting_scale_set
function satellite_dynamical_friction_acceleration_chandrasekhar
function satellite_tidal_heating_rate_gnedin
function tidal_radius_heated_halo_solver
function starFormation_rate_surface_density_disk_kmt09
subroutine starFormation_rate_surface_density_disks_exschmidt_initialize

function radiation_intergalactic_background_internal_update
subroutine satellite_orbit_extremum_phase_space_coordinates
function satellite_tidal_stripping_rate_zentner2005
function starFormation_feedback_disk_outflow_rate_creasey2012_integrand
subroutine starFormation_rate_surface_density_disks ks_initialize

file: numerical.constants.units.F90
Description: Contains a module of useful unit conversions.
Code lines: 45

module: numerical_constants_units
Description: Contains various useful unit conversions.
Code lines: 25
Contained by: file numerical.constants.units.F90
Modules used: numerical_constants_math numerical_constants_prefixes
Used by: program xray_absorption_iswilm2000 function atomic_cross_section_ionization_photo_verner
function atomic_rate_ionization_collisonal_verner subroutine chemical_hydrogen_rate_h2_gamma_to_2h
subroutine chemical_hydrogen_rate_h2_gamma_to_h2plus_electron subroutine chemical_hydrogen_rate_h2_h_to_3h
subroutine chemical_hydrogen_rate_h2plus_gamma_to_2hplus_electron subroutine chemical_hydrogen_rate_h_plusgamma_to_2hplus
subroutine chemical_hydrogen_rate_hminus_gamma_to_h_electron subroutine chemical_hydrogen_rate_hminus_h_to_2h_electron
subroutine chemical_hydrogen_rate_hminus_hplus_to_h2plus_electron subroutine chemical_hydrogen_rate_h_minus_hplus_h function cross_section_h2_minus_hplus
function cross_section_h2plus_gamma_to_2h function cross_section_h2plus_gamma_to_hplus
function cross_section_h2plus_gamma_to_h2plus function cross_section_h2_gamma_to_h2plus
function cross_section_h2_gamma_to_h2plus
function h_minus_h2_plus_electron_rate_coefficient
19.1. Program units

subroutine cooling_function_cmb_compton subroutine intergalactic_medium_state_-internalinitialize
function intergalactic_medium_state_-internal_odes function intergalactic_medium_state_-internal_update
module numerical_constants_-astronomical module numerical_constants_atomic
function radiation_integrate_over-_cross_section subroutine radiation_flux_cmb
function backgroundradiationnodes subroutine radiation_igb_internal_flux
function radiation_intergalactic_-background_internal_update function blackbody_emission

file: numerical.fftw3.F90
Description: Contains a module which imports the FFTW3 library Fortran interface.
Code lines: 44

module: fftw3
Description: Imports the FFTW3 library Fortran interface.
Code lines: 24
Contained by: file numerical.fftw3.F90
Modules used: iso_c_binding
Used by: subroutine geometry_survey_window_-functions_caputi_2011_ukidss_uds subroutine geometry_survey_window_-functions_li_white_2009_sdss
subroutine geometry_survey_window_-functions_martin_2010_alfalfa subroutine mass_function_covariance_-matrix

function: fftw_wavenumber
Description: Return the wavenumber (in units of 1/L where L is the box length) corresponding to element k out of n of a 1-D FFT using the FFTW convention.
Code lines: 14
Contained by: module fftw3

file: numerical.integration.F90
Description: Contains a module which performs numerical integration.
Code lines: 139

module: numerical_integration
Description: Implements numerical integration.
Code lines: 119
Contained by: file numerical.integration.F90
Modules used: fgs1
Used by: program conditional_mass_function subroutine matterlambdamakedistancetable
subroutine matterlambdamakedistancetable subroutine einastoenergytablemake
subroutine einastoenergytablemake subroutine einastofourierprofiletablemake
subroutine einastofourierprofiletablemake subroutine einastofreefalltimescalefree
function einastofreefalltimescalefree
function nfwprofileenergy
function nfwprofileenergy
function nfwfreefalltimescalefree
function nfwfreefalltimescalefree
function galactic_structure_velocity_disparity
function galactic_structure_velocity_disparity
subroutine galacticus_output_tree_velocity_disparity
function galacticus_output_trees_line_of_sight_velocity_disparity
function galacticus_output_trees_vlcty_dsprsn_lambdar_intgrnd1
subroutine intergalacticmediumstateelectronscatteringtabulate
function intergalactic_medium_state_internal_odes
function intergalactic_medium_state_internal_update
function generalized_press_schechter_branch_mass_root
function generalized_press_schechter_branching_probability
function modified_press_schechter_branch_mass_root
function modified_press_schechter_branching_probability
subroutine merger_tree_build_initialize
function mass_distribution_mass_enc_by_sphere_spherical
function radiation_integrate_over_cross_section
function radiation_intergalactic_background_internal_update
function imf_energy_input_rate_noninstantaneous
function imf_metal_yield_rate_noninstantaneous
function star_formation_feedback_disk_outflow_rate_creasey2012
function halo_occupancy_time_integrand
function star_formation_timescale_disk_integrated_sd
function mass_function_time_integrand_i
function stellar_population_luminosity
function g_2_integrated
function halo_mass_function_integrated
program test_integration
program optimal_sampling_smf

function: integrate
Description: Integrates the supplied integrand function.
Code lines: 84
Contained by: module numerical_integration
Modules used: galacticus_error iso_c_binding

subroutine: integrate_done
Description: Frees up integration objects that are no longer required.
Code lines: 9
Contained by: module numerical_integration
19.1. Program units

**subroutine: integration_gsl_error_handler**

*Description:* Handle errors from the GSL library during integration.

*Code lines:* 8

*Contained by:* module numerical_integration

*Modules used:* iso_c_binding

**file: numerical.interpolation.2D.irregular.F90**

*Description:* Contains a module which implements a convenient interface to the BIVAR 2D interpolation on irregularly spaced points package.

*Code lines:* 130

**module: numerical_interpolation_2d_irregular**

*Description:* Implements a convenient interface to the BIVAR 2D interpolation on irregularly spaced points package.

*Code lines:* 109

*Contained by:* file numerical.interpolation.2D.irregular.F90

*Used by:* module stellar_astrophysics_file program test_interpolation_2d

**type: interp2dirregularobject**

*Code lines:* 3

*Contained by:* module numerical_interpolation_2d_irregular

**interface: interpolate_2d_irregular**

*Code lines:* 3

*Contained by:* module numerical_interpolation_2d_irregular

**function: interpolate_2d_irregular_array**

*Description:* Perform interpolation on a set of points irregularly spaced on a 2D surface.

*Code lines:* 70

*Contained by:* module numerical_interpolation_2d_irregular

*Modules used:* bivar memory_management

**function: interpolate_2d_irregular_scalar**

*Description:* Perform interpolation on a set of points irregularly spaced on a 2D surface. This version is simply a wrapper that does look up for a scalar point by calling the array-based version.

*Code lines:* 16

*Contained by:* module numerical_interpolation_2d_irregular

**file: numerical.interpolation.F90**

*Description:* Contains a module which acts as a simple interface to the GNU Scientific Library interpolation routines.

*Code lines:* 323

**module: numerical_interpolation**

*Description:* A simple interface to the GNU Scientific Library interpolation routines.

*Code lines:* 302

*Contained by:* file numerical.interpolation.F90

*Modules used:* fgsl iso_c_binding
19. Source Code Documentation

Used by:
- subroutine filedestructor
- subroutine filedestructor
- subroutine get_interpolation
- function matterdarkenergycosmictime
- function matterlambdacosmictime
- function matterlambdadistanceangular
- function matterlambdadistancecomoving
- function matterlambdadistancecomovingconvert
- function matterlambdaexpansionfactor
- subroutine matterlambdamakeexpansionfactortable
- function matterdarkenergymakeexpansionfactortable
- subroutine matterlambdadestructor
- function matterlambdacosmictime
- function matterlambdadistanceangular
- function matterlambdadistancecomoving
- function matterlambdadistancecomovingconvert
- function matterlambdaexpansionfactor
- subroutine matterlambdamakeexpansionfactortable
- subroutine matterlambdamakedistancetable
- subroutine matterlambdamakeexpansionfactortable
- subroutine matterlambdamakedistancetable
- subroutine matterlambdamakeexpansionfactortable
- subroutine matterlambdatimeatdistancecomoving
- subroutine matterlambdatimeatdistancecomoving
- subroutine einastodestructor
- subroutine einastoenergytablemake
- subroutine einastofourierprofiletablemake
- function einastofreefallradius
- function einastofreefallradiusincreaserate
- function einastokspace
- subroutine einastoradiusfromspecificangularmomentumscalefree
- subroutine einastoradiusfromspecificangularmomentumtablemake
- function einastoenergy
- function einastoenergygrowthrate
- subroutine einastoenergytablemake
- function einastofreefallradius
- function einastofreefallradiusincreaserate
- function einastokspace
- subroutine einastoradiusfromspecificangularmomentumscalefree
- subroutine einastoradiusfromspecificangularmomentumtablemake
- function filter_response
- function fileneutralheliumfraction
- function filesinglyionizedheliumfraction
- function filetemperature
- function filespectrum
- subroutine get_interpolation
function internalelectronfraction
function internalneutralheliumfraction

function internalsinglyionizedheliumfraction
function fileelectronfraction
function fileneutralhydrogenfraction

function filetemperature
function internalfilteringmass
function internalneutralhydrogenfraction
function internaltemperature

subroutine merger_tree_build_initialize
subroutine merger_tree_history_store
subroutine merger_tree_timestep_history
subroutine merger_tree_record_evolution
subroutine merger_tree_regrid_time
subroutine merger_tree_write

subroutine history_increment

function mass_distribution_sersic_density
function mass_distribution_sersic_mass_enc_by_sphere
function mass_distribution_sersic_potential

subroutine sersic_profile_tabulate
function positionpresetposition
function positionpresetvelocity
function satellitepresetnodeindex

subroutine table_generic_1d_destroy
subroutine radiation_igb_file_flux
subroutine radiation_igb_internal_flux

function imf_energy_input_rate_noninstantaneous
function imf_recycling_rate_noninstantaneous
function imf_metal_yield_rate_noninstantaneous
function sersic_profile_tabulate

subroutine stellar_tracks_interpolation_get
subroutine stellar_population_properties_rates_noninstantaneous
function critical_overdensity_mass_scaling_gradient_wdm
function excursion_sets_first_crossing_farahi_read_file
function excursion_sets_first_crossing_rate_farahi

function stellar_population_luminosity
function stellar_population_spectra_file_interpolate

function excursion_sets_first_crossing_probability_farahi
subroutine excursion_sets_first_crossing_rate_tabulate_farahi

function critical_overdensity_mass_scaling_wdm

19.1. Program units
function excursion_sets_non_crossing_rate_farahi
function excursion_sets_first_crossing_probability_zhang_hui
function excursion_sets_first_crossing_probability_zhang_hui_high
subroutine interpolate_in_wavenumber
function linear_growth_factor_logarithmic_derivative
subroutine linear_growth_factor_simple_tabulate
function halo_mass_function_differential_tinker2008
function excursion_sets_first_crossing_probability_zhang_hui_high
function power_spectrum_nonlinear_cosmicemu
function primordial_power_spectrum_logarithmic_derivative
function transfer_function_logarithmic_derivative
program test_interpolation

function: interpolate
Description: Perform an interpolation of \( x \) into \( x_{Array}() \) and return the corresponding value in \( y_{Array}() \).
Code lines: 94
Contained by: module numerical_interpolation
Modules used: galacticus_error iso_varying_string

function: interpolate_derivative
Description: Perform an interpolation of \( x \) into \( x_{Array}() \) and return the corresponding first derivative of \( y_{Array}() \).
Code lines: 69
Contained by: module numerical_interpolation
Modules used: galacticus_error iso_varying_string

subroutine: interpolate_done
Description: Free interpolation objects when they are no longer required.
Code lines: 20
Contained by: module numerical_interpolation

function: interpolate_linear_do
Description: Given an array index \( i_{Interpolate} \) and interpolating factors \( interpolationFactors \) for array \( y_{Array}() \), return a linearly interpolated value.
Code lines: 11
Contained by: module numerical_interpolation
Modules used: kind_numbers

function: interpolate_linear_generate_factors
Description: Return interpolating factors for linear interpolation in the array \( x_{Array}() \) given the index in the array which brackets value \( x \).
Code lines: 13
Contained by: module numerical_interpolation
Modules used: kind_numbers

function: interpolate_linear_generate_gradient_factors
Description: Return interpolating factors for linear interpolation in the array \( x_{Array}() \) given the index in the array which brackets value \( x \).
Code lines: 12

1312
19.1. Program units

**Contained by:** module `numerical_interpolation`

**function: interpolate_locate**

*Description:* Perform an interpolation of x into xArray() and return the corresponding value in yArray().

*Code lines:* 53

*Contained by:* module `numerical_interpolation`

*Modules used:* `galacticus_error` `kind_numbers`

**file: numerical.meshes.F90**

*Description:* Contains a module which provides tools for working with grids.

*Code lines:* 107

**module: meshes**

*Description:* Provide tools for working with grids.

*Code lines:* 87

*Contained by:* file `numerical.meshes.F90`

*Used by:* subroutine `geometry_survey_window_functions_caputi_2011_ukidss_uds` subroutine `geometry_survey_window_functions_li_white_2009_sdss`

*Modules used:* `galacticus_error` `iso_c_binding`

**subroutine: meshes_apply_point**

*Description:* Apply a point to a mesh.

*Code lines:* 60

*Contained by:* module `meshes`

*Modules used:* `galacticus_error`

**function: triangular_shaped_cloud_integral**

*Description:* Return the integral over a triangular shaped cloud given the fraction of the cloud length in a cell.

*Code lines:* 11

*Contained by:* module `meshes`

**file: numerical.random.F90**

*Description:* Contains a module which implements pseudo-random numbers.

*Code lines:* 117

**module: pseudo_random**

*Description:* Implements pseudo-random numbers.

*Code lines:* 97

*Contained by:* file `numerical.random.F90`

*Modules used:* `fgal`

*Used by:* subroutine `black_hole_binary_recoil_velocity_standard` subroutine `black_hole_binary_recoil_velocity_standard_state_store`

*Subroutine used:* subroutine `halo_spin_distribution_bett2007` subroutine `halo_spin_distribution_bett2007_state_store`
19. Source Code Documentation

subroutine halo_spin_distribution_lognormal_state_retrieve subroutine halo_spin_distribution_lognormal_state_store
subroutine geometry_survey_window_functions_caputi_2011_ukidss_uds subroutine geometry_survey_window_functions_li_white_2009_sdss
subroutine geometry_survey_window_functions_martin_2010_alfalfa subroutine merger_tree_build_cole2000_snapshot
subroutine merger_tree_build_cole2000_state_retrieve subroutine merger_tree_build_cole2000_state_store
subroutine merger_tree_build_cole2000 Cole2000 subroutine merger_tree_build_initialize
subroutine node_component_satellite_orbiting_create subroutine jiang2008statestore
subroutine jiang2008statestore subroutine

subroutine jiang2008statestore subroutine

laceycole1993tormenstatestore subroutine jiang2008statestore

subroutine laceycole1993tormenstatestore subroutine

laceycole1993tormenstatestore subroutine

function virial_orbital_parameters_benson2005 subroutine virial_orbital_parameters_benson2005_state_retrieve
function virial_orbital_parameters_benson2005_state_store subroutine virial_orbital_parameters_wetzel2010
function virial_orbital_parameters_wetzel2010_state_retrieve subroutine virial_orbital_parameters_wetzel2010_state_store subroutine

program test_random

subroutine pseudo_random_free
Description: Frees a pseudo-random sequence object.
Code lines: 7
Contained by: module pseudo_random

function pseudo_random_get
Description: Returns a scalar giving a pseudo-random number.
Code lines: 50
Contained by: module pseudo_random
Modules used: input_parameters

subroutine pseudo_random_retrieve
Description: Stores a pseudo-random sequence object to file.
Code lines: 10
Contained by: module pseudo_random

subroutine pseudo_random_store
Description: Stores a pseudo-random sequence object to file.
Code lines: 9
Contained by: module pseudo_random
file: numerical.random.gaussian.F90
Description: Contains a module which implements Gaussian random deviates.
Code lines: 91

module: gaussian_random
Description: Implements Gaussian random deviates.
Code lines: 71
Contained by: file numerical.random.gaussian.F90
Modules used: fgsl
Used by: function halo_spin_distribution_lognormal
function jiang2008timeuntilmerging
function jiang2008timeuntilmerging
function laceycole1993tormentimeuntilmerging
function jiang2008timeuntilmerging
function laceycole1993tormentimeuntilmerging

function: gaussian_random_get
Description: Returns a Gaussian random deviate.
Code lines: 47
Contained by: module gaussian_random

Modules used: input_parameters

file: numerical.random.quasi.F90
Description: Contains a module which implements quasi-random sequences.
Code lines: 105

module: quasi_random
Description: Implements quasi-random sequences.
Code lines: 85
Contained by: file numerical.random.quasi.F90
Modules used: fgsl
Used by: subroutine merger_tree_build_initialize

subroutine: quasi_random_free
Description: Frees a quasi-random sequence object.
Code lines: 7
Contained by: module quasi_random
interface: quasi_random_get  
  Code lines: 3  
  Contained by: module quasi_random

function: quasi_random_get_array  
  Description: Returns an array giving a quasi-random points in a quasiSequenceDimension-dimensional space.  
  Code lines: 29  
  Contained by: module quasi_random

function: quasi_random_get_scalar  
  Description: Returns a scalar giving a quasi-random point in a 1-dimensional space.  
  Code lines: 29  
  Contained by: module quasi_random

file: numerical.ranges.F90  
  Description: Contains a module which implements construction of numerical ranges.  
  Code lines: 66

module: numerical_ranges  
  Description: Implements construction of numerical ranges.  
  Code lines: 46  
  Contained by: file numerical.ranges.F90  
  Used by:  
  program conditional_mass_function  
  program xray_absorption_ism_wilms2000  
  subroutine matterdarkenergymakeexpansionfactortable  
  subroutine matterlambdamakedistancetable  
  subroutine einastoenergytablemakemakeexpansionfactortable  
  subroutine einastofourierprofiletablemake  
  subroutine einastoradiusfromspecificangularmomentumtablemake  
  subroutine galacticus_meta_evolver_-profile
subroutine merger_tree_build_initialize
subroutine merger_tree_timestep_history
subroutine merger_tree_timestep_record_evolution
subroutine merger_tree_timestep_regrid_time
subroutine history_combine
subroutine history_extend
subroutine history_timesteps
subroutine table_linear_id_create
program optimal_sampling_smf
subroutine radiation_intergalactic_background_internal_initialize
function imf_energy_input_rate_noninstantaneous
function imf_metal_yield_rate_noninstantaneous
function imf_recycling_rate_noninstantaneous
subroutine mass_function_covariance_matrix
function excursion_sets_first_crossing_probability_farahi
function excursion_sets_first_crossing_probability_zhang_hui
function excursion_sets_first_crossing_probability_zhang_hui_high
subroutine linear_growth_factor_simple_tabulate
subroutine transfer_function_bbks_make
subroutine transfer_function_eisenstein_hu_make
subroutine transfer_function_file_read
subroutine transfer_function_null_make
program test_make_ranges

function: make_range
Description: Builds a numerical range between rangeMinimum and rangeMaximum using rangeNumber points and spacing as specified by rangeType (defaulting to linear spacing if no rangeType is given).
Code lines: 33
Contained by: module numerical_ranges
Modules used: galacticus_error

file: numerical.root_finder.F90
Description: Contains a module which does root finding.
Code lines: 375

module: root_finder
Description: Implements root finding.
Code lines: 355
Contained by: file numerical.root_finder.F90
Modules used: fgs1 iso_c_binding
Used by: function black_hole_binary_initial_radius_tidal_radius
         function cooling_radius_simple
         function matterdarkenergydominationepochmatter
         function matterdarkenergyequalityepochmatterdarkenergy
19. Source Code Documentation

function  
matterdarkenergydominationepochmatter  
function einastocircularvelocitymaximum  
function nfw1996concentration  
function dark_matter_profile_scale  
function galactic_structure_radius_.enclosing_mass  
function galactic_structure_radius_.initial_adiabatic  
function hot_halo_ram_pressure_.stripping_font2008.get  
function generalized_press_schechter_.branch_mass  
function modified_press_schechter_.branch_mass  
function assign_scale_raddi  
function virial_orbital_parameters_.wetzel2010  
function sersic_profile_tabulate  
function perturbation_maximum_radius  
program test_root_finding

function: root_finder_find  
Description: Finds the root of the supplied root function.  
Code lines: 153  
Contained by: module root_finder  
Modules used: galacticus_display  
galacticus_error  
is_varying_string

function: root_finder_is_initialized  
Description: Return whether a rootFinder object is initialized.  
Code lines: 7  
Contained by: module root_finder

subroutine: root_finder_range_expand  
Description: Sets the rules for range expansion to use in a rootFinder object.  
Code lines: 34  
Contained by: module root_finder  
Modules used: galacticus_error

subroutine: root_finder_root_function  
Description: Sets the function to use in a rootFinder object.  
Code lines: 9  
Contained by: module root_finder

subroutine: root_finder_tolerance  
Description: Sets the tolerances to use in a rootFinder object.  
Code lines: 9  
Contained by: module root_finder
subroutine: root_finder_type
Description: Sets the type to use in a rootFinder object.
Code lines: 10
Contained by: module root_finder

function: root_finder_wrapper_function
Description: Wrapper function callable by FGSL used in root finding.
Code lines: 9
Contained by: module root_finder

type: rootfinder
Description: Type containing all objects required when calling the FGSL root solver function.
Code lines: 66
Contained by: module root_finder

file: numerical_search.F90
Description: Contains a module which implements searching of ordered arrays.
Code lines: 252

module: arrays_search
Description: Implements searching of ordered arrays.
Code lines: 231
Contained by: file numerical_search.F90
Modules used: iso_c_binding
Used by: function gaunt_factor_sutherland1998 subroutine galacticus_meta_evolver_profile subroutine star Formation history profile subroutine record metallicity split subroutine galacticus_next_output_time subroutine galacticus_output_time_index subroutine galacticus_previous_output_time subroutine intergalactic medium state internal update subroutine descendent node sort index subroutine merger tree read do subroutine sussing tree indices read subroutine history combine subroutine radiation intergalactic background internal update subroutine set integer scalar vs subroutine delete integer scalar vs subroutine value integer scalar vs

interface: search_array
Description: Generic interface for array searching routines.
Code lines: 5
Contained by: module arrays_search

function: search_array_double
Description: Searches an array, \( x = (\text{arrayToSearch}) \), for value, \( v(=\text{valueToFind}) \), to find the index \( i \) such that \( x(i) \leq v < x(i + 1) \).
Code lines: 11
Contained by: module arrays_search
**Modules used:** fgsl

**function:** search_array_for_closest

*Description:* Searches an array, \( x = (\text{arrayToSearch}) \), for the entry closest to value, \( v(=\text{valueToFind}) \) and returns the index of that element in the array. Optionally, a tolerance may be specified within which the two values must match.

*Code lines:* 31
*Contained by:* module arrays_search
*Modules used:* fgsl, galacticus_error

**Function:** search_array_integer8

*Description:* Searches a long integer array, \( x = (\text{arrayToSearch}) \), for value, \( v(=\text{valueToFind}) \), to find the index \( i \) such that \( x(i) \leq v < x(i+1) \).

*Code lines:* 44
*Contained by:* module arrays_search
*Modules used:* kind_numbers

**Function:** search_array_varstring

*Description:* Searches an array, \( x = (\text{arrayToSearch}) \), for value, \( v(=\text{valueToFind}) \), to find the index \( i \) such that \( x(i) = v \). With this algorithm, if multiple elements of \( x() \) have the same value, then the largest value of \( i \) for which \( x(i) = v \) occurs will be returned.

*Code lines:* 46
*Contained by:* module arrays_search
*Modules used:* iso_varying_string

**Interface:** search_indexed

*Description:* Generic interface for array searching routines using indexing.

*Code lines:* 3
*Contained by:* module arrays_search

**Function:** search_indexed_integer8

*Description:* Searches a long integer array, \( x = (\text{arrayToSearch}) \), which is rank ordered when indexed by \( \text{arrayIndex} \), for value, \( v(=\text{valueToFind}) \), to find the index \( i \) such that \( x(i) \leq v < x(i+1) \).

*Code lines:* 67
*Contained by:* module arrays_search
*Modules used:* iso_c_binding, kind_numbers

**File:** numerical.sort.F90

*Description:* Contains a module which implements sorting sequences.

*Code lines:* 248

**Module:** sort

*Description:* Implements sorting.

*Code lines:* 228
*Contained by:* file numerical.sort.F90
*Modules used:* fgsl, iso_c_binding

*Used by:* subroutine output_times_initialize, subroutine merger_tree_build_initialize, subroutine create_node_indices, subroutine galacticustreeindicesread
19.1. Program units

subroutine sussingtreeindicesread subroutine sussingtreeindicesread
subroutine galacticustreeindicesread subroutine merger_tree_regrid_time
subroutine merger_tree_write program test_sort

function: compare_double
Description: Comparison function for double precision data.
Code lines: 16
Contained by: module sort

function: compare_integer
Description: Comparison function for integer data.
Code lines: 16
Contained by: module sort

function: compare_integer8
Description: Comparison function for integer data.
Code lines: 16
Contained by: module sort

interface: sort_do
Description: Generic interface to in-place sort routines.
Code lines: 6
Contained by: module sort

subroutine: sort_do_double
Description: Given an unsorted double precision array, sorts it in place.
Code lines: 7
Contained by: module sort

subroutine: sort_do_double_c
Description: Do a double precision sort.
Code lines: 12
Contained by: module sort

subroutine: sort_do_integer
Description: Given an unsorted integer array, sorts it in place.
Code lines: 7
Contained by: module sort

subroutine: sort_do_integer8
Description: Given an unsorted long integer array, sorts it in place.
Code lines: 8
Contained by: module sort
Modules used: kind_numbers

subroutine: sort_do_integer8_both
Description: Given an unsorted long integer array, sorts it in place while also rearranging array2 in the same way.
Code lines: 19
Contained by: module sort
19. Source Code Documentation

Modules used: kind_numbers

subroutine: sort_do_integer8_c
Description: Do a long integer sort.
Code lines: 13
Contained by: module sort
Modules used: kind_numbers

subroutine: sort_do_integer_c
Description: Do a integer sort.
Code lines: 12
Contained by: module sort

interface: sort_index_do
Description: Generic interface to index sort routines.
Code lines: 4
Contained by: module sort

function: sort_index_do_double
Description: Given an unsorted double array, sorts it in place.
Code lines: 10
Contained by: module sort
Modules used: kind_numbers

subroutine: sort_index_do_double_c
Description: Do a integer sort.
Code lines: 15
Contained by: module sort
Modules used: kind_numbers

function: sort_index_do_integer8
Description: Given an unsorted integer array, sorts it in place.
Code lines: 10
Contained by: module sort
Modules used: kind_numbers

subroutine: sort_index_do_integer8_c
Description: Do a integer sort.
Code lines: 15
Contained by: module sort
Modules used: kind_numbers

file: objects.abundances.F90
Description: Contains a module which defines the abundances structure used for describing elemental abundances in GALACTICUS.
Code lines: 977

module: abundances_structure
Description: Defines the abundances structure used for describing elemental abundances in GALACTICUS.
19.1. Program units

Code lines: 957
Contained by: file objects.abundances.F90
Modules used: iso_varying_string
Used by: subroutine chemical_densities_cie_file_interpolate
        function electron_density_cie_file_interpolate
        subroutine chemical_densities_cie_file
        function electron_density_cie_file
        subroutine chemical_densities_density_log_slope_cie_file
module chemical_states

subroutine chemical_state_atomic_cie_cloudy_create
function electron_density_density_log_slope_atomic_cie_cloudy
subroutine cooling_function_cie_file

function cooling_function_cie_file_logtemperature_interpolate
subroutine cooling_function_temperature_slope_cie_file
subroutine cooling_function_density_slope_cie_file
module cooling_functions

subroutine cooling_function_atomic_cie_cloudy_create
subroutine cooling_function_temperature_slope_atomic_cie_cloudy
subroutine cooling_function_molecular_temperature_slope_molecular_hydrogen_gp
module cooling_radii_isothermal
module cooling_times

function cooling_time_simple
module galacticus_output_star_formation_histories
subroutine star_formation_history_record_metallicity_split
subroutine star_formation_history_record_null
module intergalactic_medium_state_internal_evolver
module galacticus_nodes

subroutine node_component_disk_-exponential_initialize
subroutine node_component_disk_-exponential_post_evolve
subroutine node_component_disk_-exponential_satellite_merging
subroutine node_component_hot_halo_-cold_mode_formation
subroutine node_component_hot_halo_-cold_mode_node_merger
subroutine node_component_hot_halo_-cold_mode_push_to_cooling_pipes
subroutine node_component_hot_halo_-cold_mode_scale_set
subroutine node_component_hot_halo_-cold_mode_tree_initialize
subroutine node_component_hot_halo_-outflow_tracking_rate_compute
subroutine node_component_hot_halo_-outflow_tracking_scale_set
subroutine node_component_hot_halo_-standard_formation
subroutine node_component_hot_halo_-standard_node_merger
subroutine node_component_hot_halo_-standard_outflow_return
subroutine node_component_hot_halo_-standard_post_evolve
subroutine node_component_hot_halo_-standard_push_to_cooling_pipes
subroutine node_component_hot_halo_-standard_satellite_merger
subroutine node_component_hot_halo_-standard_tree_initialize
subroutine node_component_spheroid_-standard_initialize
subroutine node_component_spheroid_-standard_post_evolve
subroutine node_component_spheroid_-standard_satellite_merging
subroutine stellar_luminosities_set

module star_formation_imf
function imf_select_fixed

function star_formation_rate_surface_-density_disk_br
function star_formation_rate_surface_-density_disk_kmt09
function star_formation_rate_surface_-density_disk_exschmidt

module radiation_intergalactic_-background_internal
function imf_select_disk_spheroid
function star_formation_rate_surface_-density_disk
function star_formation_rate_surface_-density_disk_k
module stellar_population_luminosities
module stellar_population_properties subroutine stellar_population_properties_rates_instantaneous

subroutine stellar_population_propertiesRates_instantaneous subroutine stellar_population_properties_rates_noninstantaneous subroutine stellar_population_properties_scales_instantaneous subroutine stellar_population_properties_scales_noninstantaneous subroutine stellar_population_properties_noninstantaneous_initialize subroutine stellar_population_properties_rates_noninstantaneous subroutine stellar_population_properties_scales_noninstantaneous

function stellar_population_spectra_conroy_get function stellar_population_spectra_file_get function stellar_population_spectra_file_interpolate module stellar_population_spectra

program test_abundances

type: abundances

description: The abundances structure used for describing elemental abundances in GALACTICUS.
code lines: 195
contained by: module abundances_structure

function: abundances_add
description: Add two abundances objects.
code lines: 17
contained by: module abundances_structure

subroutine: abundances.allocate_elemental_values
description: Ensure that the elementalValue array in an abundances is allocated.
code lines: 8
contained by: module abundances_structure
modules used: memory_management

function: abundances.atomic_index
description: Return the atomic index for the specified entry in the abundances structure.
code lines: 22
contained by: module abundances_structure
modules used: galacticus_error

subroutine: abundances.builder
description: Build a abundances object from the given XML abundancesDefinition.
code lines: 29
contained by: module abundances_structure
modules used: fox_dom galacticus_error

subroutine: abundances.deserialize
description: Pack abundances from an array into an abundances structure.
code lines: 19
contained by: module abundances_structure

subroutine: abundances.destroy
description: Destroy an abundances object.
code lines: 8

1325
19. Source Code Documentation

**Contained by:** module abundances_structure

**Modules used:** memory_management

**function:** abundances_divide

**Description:** Divide an abundances object by a scalar.

**Code lines:** 12

**Contained by:** module abundances_structure

**subroutine:** abundances_dump

**Description:** Reset an abundances object.

**Code lines:** 23

**Contained by:** module abundances_structure

**Modules used:** galacticus_display

**subroutine:** abundances_dump_raw

**Description:** Dump an abundances object to binary.

**Code lines:** 12

**Contained by:** module abundances_structure

**function:** abundances_get_metallicity

**Description:** Return the metallicity of the self structure.

**Code lines:** 40

**Contained by:** module abundances_structure

**Modules used:** galacticus_error

**function:** abundances_helium_mass_fraction

**Description:** Returns the mass fraction of helium.

**Code lines:** 13

**Contained by:** module abundances_structure

**function:** abundances_helium_number_fraction

**Description:** Returns the mass fraction of helium.

**Code lines:** 16

**Contained by:** module abundances_structure

**function:** abundances_hydrogen_mass_fraction

**Description:** Returns the mass fraction of hydrogen.

**Code lines:** 14

**Contained by:** module abundances_structure

**function:** abundances_hydrogen_number_fraction

**Description:** Returns the number fraction of hydrogen.

**Code lines:** 16

**Contained by:** module abundances_structure

**subroutine:** abundances_increment

**Description:** Increment an abundances object.

**Code lines:** 12

**Contained by:** module abundances_structure
subroutine: abundances_initialize
Description: Initialize the abundanceStructure object module. Determines which abundances are to be tracked.
Code lines: 48
Contained by: module abundances_structure
Modules used: atomic_data input_parameters memory_management

function: abundances_is_zero
Description: Test whether an abundances object is zero.
Code lines: 15
Contained by: module abundances_structure

subroutine: abundances_mass_to_mass_fraction
Description: Convert abundance masses to mass fractions by dividing by mass while ensuring that the fractions remain within the range 0–1.
Code lines: 15
Contained by: module abundances_structure

subroutine: abundances_mass_to_mass_fraction_packed
Description: Convert abundance masses to mass fractions by dividing by mass while ensuring that the fractions remain within the range 0–1.
Code lines: 29
Contained by: module abundances_structure

function: abundances_max
Description: Return an element-by-element max() on two abundances objects.
Code lines: 9
Contained by: module abundances_structure

function: abundances_multiply
Description: Multiply an abundances object by a scalar.
Code lines: 12
Contained by: module abundances_structure

function: abundances_multiply_switched
Description: Multiply a scalar by an abundances object.
Code lines: 9
Contained by: module abundances_structure

function: abundances_names
Description: Return a name for the specified entry in the abundances structure.
Code lines: 23
Contained by: module abundances_structure
Modules used: galacticus_error

subroutine: abundances_output
Description: Store an abundances object in the output buffers.
Code lines: 16
19. Source Code Documentation

- **abundances_output_count**
  - **Description:** Increment the output count to account for an abundances object.
  - **Code lines:** 9
  - **Contained by:** module abundances_structure

- **abundances_output_names**
  - **Description:** Assign names to output buffers for an abundances object.
  - **Code lines:** 25
  - **Contained by:** module abundances_structure

- **abundances_property_count**
  - **Description:** Return the number of properties required to track abundances. This is equal to the number of elements tracked, elementsCount, plus one since we always track a total metallicity.
  - **Code lines:** 10
  - **Contained by:** module abundances_structure

- **abundances_read_raw**
  - **Description:** Read an abundances object from binary.
  - **Code lines:** 12
  - **Contained by:** module abundances_structure

- **abundances_reset**
  - **Description:** Reset an abundances object.
  - **Code lines:** 11
  - **Contained by:** module abundances_structure

- **abundances_serialize**
  - **Description:** Pack abundances from an array into an abundances structure.
  - **Code lines:** 18
  - **Contained by:** module abundances_structure

- **abundances_set_metallicity**
  - **Description:** Set the metallicity of the self structure to metallicity.
  - **Code lines:** 73
  - **Contained by:** module abundances_structure

- **abundances_set_to_unity**
  - **Description:** Set an abundances object to unity.
  - **Code lines:** 13
  - **Contained by:** module abundances_structure

- **abundances_subtract**
  - **Description:** Subtract two abundances objects.
  - **Code lines:** 17
  - **Contained by:** module abundances_structure
interface: max
Code lines: 2
Contained by: module abundances_structure

interface: operator(\*)
Code lines: 2
Contained by: module abundances_structure

file: objects.chemical_abundances.F90
Description: Contains a module which defines the structure used for describing chemical abundances in GALACTICUS.
Code lines: 645

module: chemical_abundances_structure
Description: Defines the structure used for describing chemical abundances in GALACTICUS.
Code lines: 625
Contained by: file objects.chemical_abundances.F90
Modules used: iso_varying_string
Used by:
- module accretion_halos
  - function coldmodeaccretionratechemicals
  - function coldmodeaccretedmasschemicals
- function coldmodecoldmodefraction
- function nullaccretionratechemicals
- function nullaccretedmasschemicals
- function simpleaccretionratechemicals
- function simpleaccretedmasschemicals
- function simpleconstructor
- function coldmodecoldmodefraction
- function nullaccretedmasschemicals
- function simpleaccretionratechemicals
- function simpleaccretedmasschemicals
- module chemical_hydrogen_rates
- subroutine chemical_densities_cie_file
- subroutine chemical_state_cie_
  - chemicals_initialize
- subroutine chemical_densities_atomic_
  - cie_cloudy
- subroutine cooling_function_density_
  - slope_cie_file
- subroutine cooling_function_cmb_compton
- subroutine cooling_function_
  - temperature_slope_cie_file
- subroutine cooling_function_cmb_compton
- subroutine cooling_function_atomic_
  - cie_cloudy
- subroutine cooling_function_density_
  - slope_cie_file
  - slope_atomic_cie_cloudy

Subroutine chemical_reaction_rates_null_compute
Subroutine chemical_densities_cie_file_interpolate
Module chemical_states
Subroutine chemical_reaction_rates
Subroutine chemical_densities_rates
Subroutine chemical_densities_rates
subroutine cooling_function_temperature_slope_atomic_cie_cloudy subroutine cooling_function_density_slope_molecular_hydrogen_gp
function cooling_function_gp_h2plus_electron function cooling_function_gp_h_h2plus
function cooling_function_gp_h_h2plus
subroutine cooling_function_molecular_hydrogen_gp_initialize subroutine cooling_function_temperature_slope_molecular_hydrogen_gp
module cooling_radii_isothermal module cooling_radii_simple
module cooling_times function cooling_time_density_log_slope_simple
function cooling_time_simple function cooling_time_temperature_log_slope_simple
module galacticus_nodes subroutine node_component_hot_halo_outflow_tracking_scale_set
subroutine node_component_hot_halo_standard_formation subroutine node_component_hot_halo_standard_initialize
subroutine node_component_hot_halo_standard_node_merger subroutine node_component_hot_halo_standard_outflow_return
subroutine node_component_hot_halo_standard_push_from_halo subroutine node_component_hot_halo_standard_rate_compute
subroutine node_component_hot_halo_standard_satellite_merger subroutine node_component_hot_halo_standard_scale_set
subroutine node_component_hot_halo_standard_tree_initialize

function: chemical_abundances_add
Description: Add two abundances objects.
Code lines: 19
Contained by: module chemical_abundances_structure

subroutine: chemical_abundances_allocate_values
Description: Ensure that the chemicalValue array in an chemicalsStructure is allocated.
Code lines: 14
Contained by: module chemical_abundances_structure
Modules used: memory_management

subroutine: chemical_abundances_deserialize
Description: Pack abundances from an array into an abundances structure.
Code lines: 17
Contained by: module chemical_abundances_structure

function: chemical_abundances_divide
Description: Divide a chemical abundances object by a scalar.
Code lines: 15
Contained by: module chemical_abundances_structure
subroutine: chemical_abundances_increment
Description: Increment an abundances object.
Code lines: 10
Contained by: module chemical_abundances_structure

subroutine: chemical_abundances_initialize
Description: Initialize the chemicalAbundanceStructure object module. Determines which chemicals are to be tracked.
Code lines: 55
Contained by: module chemical_abundances_structure
Modules used: chemical_structures input_parameters memory_management

function: chemical_abundances_multiply
Description: Multiply a chemical abundances object by a scalar.
Code lines: 15
Contained by: module chemical_abundances_structure

subroutine: chemical_abundances_serialize
Description: Pack abundances from an array into an abundances structure.
Code lines: 16
Contained by: module chemical_abundances_structure

function: chemical_abundances_subtract
Description: Subtract two abundances objects.
Code lines: 19
Contained by: module chemical_abundances_structure

type: chemicalabundances
Description: The structure used for describing chemical abundances in GALACTICUS.
Code lines: 159
Contained by: module chemical_abundances_structure

function: chemicals_abundances
Description: Returns the abundance of a molecule in the chemical abundances structure given the moleculeIndex.
Code lines: 8
Contained by: module chemical_abundances_structure

subroutine: chemicals_abundances_destroy
Description: Destroy a chemical abundances object.
Code lines: 8
Contained by: module chemical_abundances_structure
Modules used: memory_management

function: chemicals_abundances_is_zero
Description: Test whether an chemicals object is zero.
Code lines: 9
Contained by: module chemical_abundances_structure
subroutine: chemicals_abundances_reset
Description: Resets all chemical abundances to zero.
Code lines: 14
Contained by: module chemical_abundances_structure

subroutine: chemicals_abundances_set
Description: Sets the abundance of a molecule in the chemical abundances structure given the moleculeIndex.
Code lines: 15
Contained by: module chemical_abundances_structure

subroutine: chemicals_abundances_set_to_unity
Description: Resets all chemical abundances to unity.
Code lines: 14
Contained by: module chemical_abundances_structure

subroutine: chemicals_builder
Description: Build a chemicalAbundances object from the given XML chemicalsDefinition.
Code lines: 10
Contained by: module chemical_abundances_structure
Modules used: fox_dom galacticus_error

subroutine: chemicals_dump
Description: Dump all chemical values.
Code lines: 17
Contained by: module chemical_abundances_structure
Modules used: galacticus_display

subroutine: chemicals_dump_raw
Description: Dump all chemical values in binary.
Code lines: 9
Contained by: module chemical_abundances_structure

subroutine: chemicals_enforce_positive
Description: Force all chemical values to be positive, by truncating negative values to zero.
Code lines: 11
Contained by: module chemical_abundances_structure

function: chemicals_index
Description: Returns the index of a chemical in the chemical abundances structure given the chemicalName.
Code lines: 14
Contained by: module chemical_abundances_structure

subroutine: chemicals_mass_to_number
Description: Divide all chemical species by their mass in units of the atomic mass. This converts abundances by mass into abundances by number.
Code lines: 18
Contained by: module chemical_abundances_structure
function: chemicals_names
Description: Return a name for the specified entry in the chemicals structure.
Code lines: 16
Contained by: module chemical_abundances_structure
Modules used: galacticus_error

subroutine: chemicals_number_to_mass
Description: Multiply all chemical species by their mass in units of the atomic mass. This converts abundances by number into abundances by mass.
Code lines: 18
Contained by: module chemical_abundances_structure

function: chemicals_property_count
Description: Return the number of properties required to track chemicals. This is equal to the number of chemicals tracked, chemicalsCount.
Code lines: 10
Contained by: module chemical_abundances_structure

subroutine: chemicals_read_raw
Description: Read all chemical values in binary.
Code lines: 14
Contained by: module chemical_abundances_structure
Modules used: memory_management

file: objects.chemical_structure.F90
Description: Contains a module which implements structures that describe chemicals.
Code lines: 299

module: chemical_structures
Description: Implements structures that describe chemicals.
Code lines: 279
Contained by: file objects.chemical_structure.F90
Modules used: iso_varying_string numerical_constants_atomic numerical_constants_physical
Used by: subroutine chemical_abundances_-initialize

type: atomicbond
Description: A type that defines an atomic bond within a chemical.
Code lines: 3
Contained by: module chemical_structures

type: atomicstructure
Description: A type that defines an atom within a chemical.
Code lines: 5
Contained by: module chemical_structures

subroutine: chemical_database_get
19. Source Code Documentation

**Description:** Find a chemical in the database and return it.
**Code lines:** 11
**Contained by:** module chemical_structures

**function:** chemical_database_get_index
**Description:** Find a chemical in the database and return it.
**Code lines:** 19
**Contained by:** module chemical_structures
**Modules used:** galacticus_error

**function:** chemical_structure_charge
**Description:** Return the charge on a chemical.
**Code lines:** 7
**Contained by:** module chemical_structures

**subroutine:** chemical_structure_export
**Description:** Export a chemical structure to a chemical markup language (CML) file.
**Code lines:** 70
**Contained by:** module chemical_structures
**Modules used:** fox_wxml

**subroutine:** chemical_structure_initialize
**Description:** Initialize the chemical structure database by reading the atomic structure database. Note: this implementation is not fully compatible with chemical markup language (CML), but only a limited subset of it.
**Code lines:** 78
**Contained by:** module chemical_structures
**Modules used:** fox_dom galacticus_error galacticus_input_paths

**function:** chemical_structure_mass
**Description:** Return the mass of a chemical.
**Code lines:** 7
**Contained by:** module chemical_structures

**type:** chemicalstructure
**Description:** A type that defines a chemical.
**Code lines:** 41
**Contained by:** module chemical_structures

**file:** objects.coordinates.F90
**Description:** Contains a module which implements the coordinates class.
**Code lines:** 525

**module:** coordinates
**Description:** Implements the coordinates class.
**Code lines:** 505
**Contained by:** file objects.coordinates.F90
**Used by:** function betaprofiledensity function betaprofiledensitylogslope
function betaprofiledensity
function mass_distribution_beta_profile_density
function mass_distribution_beta_profile_potential
function mass_distribution_density_null
function mass_distribution_hernquist_density
function mass_distribution_hernquist_potential
function mass_distribution_mass_enc_by_sphere_spherical_integrand
function mass_distribution_nfw_density
function mass_distribution_sersic_density
function mass_distribution_sersic_potential
function node_component_spheroid_standard_density
function node_component_spheroid_standard_potential
function node_component_hot_halo_cold_mode_density_task

interface: assignment(=)
  Code lines: 4
  Contained by: module coordinates

type: coordinate
  Description: The base coordinate object class.
  Code lines: 21
  Contained by: module coordinates

type: coordinatecartesian
  Description: A Cartesian coordinate object class.
  Code lines: 50
  Contained by: module coordinates

type: coordinatecylindrical
  Description: A cylindrical coordinate object class.
  Code lines: 50
  Contained by: module coordinates

subroutine: coordinates_assign
  Description: Assign one coordinate object to another, automatically handling the conversion between coordinate systems.
  Code lines: 11
  Contained by: module coordinates

subroutine: coordinates_assign_from
  Description: Return a 3-component vector from a coordinate object.
  Code lines: 8
  Contained by: module coordinates

subroutine: coordinates_assign_to
19. Source Code Documentation

subroutine: coordinates_cartesian_from_cartesian
Description: Create a Cartesian coordinate object from a Cartesian vector.
Code lines: 8
Contained by: module coordinates

subroutine: coordinates_cartesian_set_x
Description: Return the x-component of a Cartesian coordinate object.
Code lines: 8
Contained by: module coordinates

subroutine: coordinates_cartesian_set_y
Description: Return the y-component of a Cartesian coordinate object.
Code lines: 8
Contained by: module coordinates

subroutine: coordinates_cartesian_set_z
Description: Return the z-component of a Cartesian coordinate object.
Code lines: 8
Contained by: module coordinates

function: coordinates_cartesian_to_cartesian
Description: Return a Cartesian vector from a Cartesian coordinate object.
Code lines: 8
Contained by: module coordinates

function: coordinates_cartesian_x
Description: Return the x-component of a Cartesian coordinate object.
Code lines: 7
Contained by: module coordinates

function: coordinates_cartesian_y
Description: Return the y-component of a Cartesian coordinate object.
Code lines: 7
Contained by: module coordinates

function: coordinates_cartesian_z
Description: Return the z-component of a Cartesian coordinate object.
Code lines: 7
Contained by: module coordinates

subroutine: coordinates_cylindrical_from_cartesian
Description: Create a cylindrical coordinate object from a Cartesian vector.
Code lines: 12
Contained by: module coordinates
19.1. Program units

**function:** coordinates_cylindrical_phi
*Description:* Return the \( \phi \)-component of a Cylindrical coordinate object.
*Code lines:* 7
*Contained by:* module coordinates

**function:** coordinates_cylindrical_r
*Description:* Return the \( r \)-component of a Cylindrical coordinate object.
*Code lines:* 7
*Contained by:* module coordinates

**subroutine:** coordinates_cylindrical_set_phi
*Description:* Return the \( \phi \)-component of a Cylindrical coordinate object.
*Code lines:* 8
*Contained by:* module coordinates

**subroutine:** coordinates_cylindrical_set_r
*Description:* Return the \( r \)-component of a Cylindrical coordinate object.
*Code lines:* 8
*Contained by:* module coordinates

**subroutine:** coordinates_cylindrical_set_z
*Description:* Return the \( z \)-component of a Cylindrical coordinate object.
*Code lines:* 8
*Contained by:* module coordinates

**function:** coordinates_cylindrical_to_cartesian
*Description:* Return a Cartesian vector from a cylindrical coordinate object.
*Code lines:* 12
*Contained by:* module coordinates

**function:** coordinates_cylindrical_z
*Description:* Return the \( z \)-component of a Cylindrical coordinate object.
*Code lines:* 7
*Contained by:* module coordinates

**subroutine:** coordinates_null_from
*Description:* Set generic coordinate object from Cartesian point. Simply quits with an error.
*Code lines:* 9
*Contained by:* module coordinates
*Modules used:* galacticus_error

**function:** coordinates_null_to
*Description:* Convert generic coordinate object to Cartesian point. Simply quits with an error.
*Code lines:* 9
*Contained by:* module coordinates
*Modules used:* galacticus_error
subroutine: coordinates_spherical_from_cartesian
Description: Create a spherical coordinate object from a Cartesian vector.
Code lines: 18
Contained by: module coordinates

function: coordinates_spherical_phi
Description: Return the $\phi$-component of a Spherical coordinate object.
Code lines: 7
Contained by: module coordinates

function: coordinates_spherical_r
Description: Return the $r$-component of a Spherical coordinate object.
Code lines: 7
Contained by: module coordinates

subroutine: coordinates_spherical_set_phi
Description: Return the $\phi$-component of a Spherical coordinate object.
Code lines: 8
Contained by: module coordinates

subroutine: coordinates_spherical_set_r
Description: Return the $r$-component of a Spherical coordinate object.
Code lines: 8
Contained by: module coordinates

subroutine: coordinates_spherical_set_theta
Description: Return the $\theta$-component of a Spherical coordinate object.
Code lines: 8
Contained by: module coordinates

function: coordinates_spherical_theta
Description: Return the $\theta$-component of a Spherical coordinate object.
Code lines: 7
Contained by: module coordinates

function: coordinates_spherical_to_cartesian
Description: Return a Cartesian vector from a spherical coordinate object.
Code lines: 12
Contained by: module coordinates

type: coordinatespherical
Description: A spherical coordinate object class.
Code lines: 50
Contained by: module coordinates

file: objects.history.F90
Description: Contains a module defining the history object type.
Code lines: 1253
module: histories
Description: Defines the history object type.
Code lines: 1233
Contained by: file objects.history.F90
Modules used: kind_numbers
Used by:
  subroutine galacticus_output_tree_satellite_status
  subroutine star_formation_history_create_metallicity_split
  subroutine star_formation_history_output_metallicity_split
  subroutine star_formation_history_scales_metallicity_split
  subroutine star_formation_history_output_null
  subroutine star_formation_history_scales_null
  subroutine galacticus_state_retrieve
  subroutine build_subhalo_mass_histories
  function satellitepresetmergeboundmass
  subroutine node_component_disk_exponential_create
  subroutine node_component_disk_exponential_post_evolve
  subroutine node_component_disk_exponential_rate_compute
  subroutine node_component_disk_exponential_scale_set
  module node_component_spheroid_standard
  subroutine stellar_population_properties_history_create_instantaneous
  subroutine stellar_population_properties_scales_instantaneous
  subroutine stellar_population_properties_rates_noninstantaneous
module galacticus_output_star_formation_histories
  subroutine star_formation_history_create_metallicity_split_make_history
  subroutine star_formation_history_record_metallicity_split
  subroutine star_formation_history_create_null
  subroutine star_formation_history_record_null
  subroutine output_times_initialize
  subroutine galacticus_state_store
  module galacticus_nodes
  function satellitepresetnodeindex
  subroutine node_component_disk_exponential_post_evolve
  subroutine node_component_disk_exponential_satellite_merging
  subroutine node_component_disk_exponential_scale_set
  subroutine node_component_disk_exponential_star_formation_history_output
  module stellar_population_properties
  subroutine stellar_population_properties_rates_instantaneous
  subroutine stellar_population_properties_scales_noninstantaneous
  subroutine stellar_population_properties_history_create_noninstantaneous
  subroutine stellar_population_properties_scales_noninstantaneous

subroutine: histories_state_retrieve
Description: Retrieve the history state from the file.
Code lines: 9
Contained by: module histories
Modules used: fgsl

subroutine: histories_state_store
Description: Write the history state to file.
Code lines: 9
-contained by: module histories

modules used: fgs1

type: history

description: The history object type.
code lines: 174
-contained by: module histories

function: history_add

description: Add two history objects.
code lines: 22
-contained by: module histories
-modules used: galacticus_error

subroutine: history_builder

description: Build a history object from the given XML historyDefinition.
code lines: 10
-contained by: module histories
-modules used: fox_dom galacticus_error

subroutine: history_clone

description: Clone a history object.
code lines: 19
-contained by: module histories
-modules used: memory_management

subroutine: history_combine

description: Combines the data in combineHistory with that in thisHistory. This function is designed for histories that track integrated quantities (such as total mass of stars formed in a time interval for example). thisHistory will be extended if necessary to span the range of combineHistory. Then, the data from combineHistory will be added to that in thisHistory by finding the fraction of each timestep in combineHistory that overlaps with each timestep in thisHistory and assuming that the corresponding fraction of the data value should be added to thisHistory.
code lines: 80
-contained by: module histories
-modules used: arrays_search galacticus_error iso_c_binding numerical_ranges

subroutine: history_create

description: Create a history object.
code lines: 36
-contained by: module histories
-modules used: galacticus_error memory_management numerical_ranges

subroutine: history_deserialize

description: Pack history from an array into a history structure.
code lines: 9
19.1. Program units

Contained by: module `histories`

**subroutine: history_destroy**
*Description:* Destroy a history.
*Code lines:* 19
*Contained by:* module `histories`
*Modules used:* `memory_management`

**function: history_divide**
*Description:* Divides history data by a double precision divisor.
*Code lines:* 13
*Contained by:* module `histories`

**subroutine: history_dump**
*Description:* Dumps a history object.
*Code lines:* 24
*Contained by:* module `histories`
*Modules used:* `galacticus_display iso_varying_string`

**subroutine: history_dump_raw**
*Description:* Dumps a history object in binary.
*Code lines:* 14
*Contained by:* module `histories`

**function: history_exists**
*Description:* Returns true if the history has been created.
*Code lines:* 7
*Contained by:* module `histories`

**subroutine: history_extend**
*Description:* Extends a history to encompass the given time range.
*Code lines:* 115
*Contained by:* module `histories`
*Modules used:* `galacticus_error iso_varying_string numerical_ranges range string_handling`

**subroutine: history_increment**
*Description:* Adds the data in `addHistory` to that in `thisHistory`. This function is designed for histories that track instantaneous rates. The rates in `addHistory` are interpolated to the times in `thisHistory` and added to the rates in `thisHistory`.
*Code lines:* 64
*Contained by:* module `histories`
*Modules used:* `fgsl galacticus_error iso_c_binding numerical_interpolation`

**function: history_is_zero**
*Description:* Test whether a history object is all zero.
*Code lines:* 10
subroutine: history_long_integer_builder
Description: Build a longIntegerHistory object from the given XML historyDefinition.
Code lines: 10
Contained by: module histories
Modules used: fox_dom galacticus_error

subroutine: history_long_integer_clone
Description: Clone a longIntegerHistory object.
Code lines: 19
Contained by: module histories
Modules used: memory_management

subroutine: history_long_integer_create
Description: Create a history object.
Code lines: 36
Contained by: module histories
Modules used: galacticus_error memory_management numerical_ranges

subroutine: history_long_integer_destroy
Description: Destroy a history.
Code lines: 19
Contained by: module histories
Modules used: memory_management

subroutine: history_long_integer_dump
Description: Dumps a history object.
Code lines: 24
Contained by: module histories
Modules used: galacticus_display iso_varying_string

subroutine: history_long_integer_dump_raw
Description: Dumps a history object in binary.
Code lines: 14
Contained by: module histories

function: history_long_integer_exists
Description: Returns true if the history has been created.
Code lines: 7
Contained by: module histories

subroutine: history_long_integer_read_raw
Description: Read a history object in binary.
Code lines: 19
Contained by: module histories
Modules used: memory_management
subroutine: history_long_integer_reset
   Description: Reset a history by zeroing all elements, but leaving the structure (and times) intact.
   Code lines: 7
   Contained by: module histories

subroutine: history_long_integer_trim
   Description: Removes outdated information from “future histories” (i.e. histories that store data for future reference). Removes all but one entry prior to the given currentTime (this allows for interpolation of the history to the current time). Optionally, the remove is done only if it will remove more than minimumPointsToRemove entries (since the removal can be slow this allows for some optimization).
   Code lines: 61
   Contained by: module histories
   Modules used: galacticus_error iso_c_binding memory_management

function: history_multiply
   Description: Multiplies history data by a double precision multiplier.
   Code lines: 13
   Contained by: module histories

function: history_multiply_switched
   Description: Multiply a scalar by an history object.
   Code lines: 9
   Contained by: module histories

subroutine: history_read_raw
   Description: Read a history object in binary.
   Code lines: 19
   Contained by: module histories
   Modules used: memory_management

subroutine: history_reset
   Description: Reset a history by zeroing all elements, but leaving the structure (and times) intact.
   Code lines: 7
   Contained by: module histories

subroutine: history_serialize
   Description: Pack history from an array into an history structure.
   Code lines: 9
   Contained by: module histories

function: history_serialize_count
   Description: Return the number of properties required to track a history.
   Code lines: 11
   Contained by: module histories
19. Source Code Documentation

**subroutine: history_set_times**
*Description:* Extend the range of history times to include the given timeEarliest and timeLatest.
*Code lines:* 8
*Contained by:* module histories

**subroutine: history_set_to_unity**
*Description:* Reset a history by zeroing all elements, but leaving the structure (and times) intact.
*Code lines:* 7
*Contained by:* module histories

**function: history_subtract**
*Description:* Subtract two history objects.
*Code lines:* 24
*Contained by:* module histories
*Modules used:* galacticus_error

**subroutine: history_timesteps**
*Description:* Return an array of time intervals in thisHistory.
*Code lines:* 26
*Contained by:* module histories
*Modules used:* memory_management numerical_ranges

**subroutine: history_trim**
*Description:* Removes outdated information from “future histories” (i.e. histories that store data for future reference). Removes all but one entry prior to the given currentTime (this allows for interpolation of the history to the current time). Optionally, the remove is done only if it will remove more than minimumPointsToRemove entries (since the removal can be slow this allows for some optimization).
*Code lines:* 61
*Contained by:* module histories
*Modules used:* galacticus_error iso_c_binding memory_management

**type: longintegerhistory**
*Description:* The history object type.
*Code lines:* 79
*Contained by:* module histories

**interface: operator(*)**
*Code lines:* 2
*Contained by:* module histories

**file: objects.kepler_orbits.F90**
*Description:* Contains a module which defines an orbit structure for use in GALACTICUS.
*Code lines:* 1029

**module: kepler_orbits**
*Description:* Defines an orbit structure for use in GALACTICUS.
function: kepler_orbits_angular_momentum
Description: Return the angular momentum for this orbit.
Code lines: 15
Contained by: module kepler_orbits

subroutine: kepler_orbits_angular_momentum_set
Description: Sets the tangential velocity to the specified value.
Code lines: 13
Contained by: module kepler_orbits

function: kepler_orbits_apocenter_radius
**19. Source Code Documentation**

*Description:* Return the apocenter radius for this orbit.
*Code lines:* 21
*Contained by:* module *kepler_orbits*

**subroutine: kepler_orbits_apocenter_radius_set**
*Description:* Sets the apocenter radius to the specified value.
*Code lines:* 13
*Contained by:* module *kepler_orbits*

**subroutine: kepler_orbits_assert_is_defined**
*Description:* Assert that an orbit is defined - quit with an error if it is not.
*Code lines:* 11
*Contained by:* module *kepler_orbits*
*Modules used:* galacticus_error

**subroutine: kepler_orbits_builder**
*Description:* Build a *keplerOrbit* object from the given XML *keplerOrbitDefinition*.
*Code lines:* 43
*Contained by:* module *kepler_orbits*
*Modules used:* fox_dom galacticus_error

**subroutine: kepler_orbits_destroy**
*Description:* Destroy an orbit.
*Code lines:* 7
*Contained by:* module *kepler_orbits*

**subroutine: kepler_orbits_dump**
*Description:* Reset an orbit to a null state.
*Code lines:* 63
*Contained by:* module *kepler_orbits*
*Modules used:* galacticus_display iso_varying_string

**subroutine: kepler_orbits_dump_raw**
*Description:* Dump a *keplerOrbit* object in binary.
*Code lines:* 18
*Contained by:* module *kepler_orbits*

**function: kepler_orbits_eccentricity**
*Description:* Return the eccentricity for this orbit.
*Code lines:* 18
*Contained by:* module *kepler_orbits*

**subroutine: kepler_orbits_eccentricity_set**
*Description:* Sets the tangential velocity to the specified value.
*Code lines:* 13
*Contained by:* module *kepler_orbits*

**function: kepler_orbits_energy**
*Description:* Return the energy for this orbit.
19.1. Program units

Code lines: 20
Contained by: module kepler_orbits
Modules used: numerical_constants_physical

subroutine: kepler_orbits_energy_set
Description: Sets the tangential velocity to the specified value.
Code lines: 13
Contained by: module kepler_orbits

function: kepler_orbits_host_mass
Description: Return the host mass for this orbit.
Code lines: 12
Contained by: module kepler_orbits
 Modules used: galacticus_error

function: kepler_orbits_is_bound
Description: Returns true if the orbit is bound.
Code lines: 13
Contained by: module kepler_orbits

function: kepler_orbits_is_defined
Description: Returns true if the orbit is fully defined. For the orbits consider here, in which we don’t care about the orientation of the orbital plane or the argument of pericenter, this requires that three orbital parameter be set (in addition to the masses of the orbitting bodies).
Code lines: 30
Contained by: module kepler_orbits

subroutine: kepler_orbits_masses_set
Description: Sets the masses of the two orbitting objects in a keplerOrbit object.
Code lines: 14
Contained by: module kepler_orbits

subroutine: kepler_orbits_output
Description: Store a keplerOrbit object in the output buffers.
Code lines: 20
Contained by: module kepler_orbits
 Modules used: kind_numbers

subroutine: kepler_orbits_output_count
Description: Increment the output count to account for a keplerOrbit object.
Code lines: 9
Contained by: module kepler_orbits

subroutine: kepler_orbits_output_names
Description: Assign names to output buffers for a keplerOrbit object.
Code lines: 29
Contained by: module kepler_orbits
 Modules used: numerical_constants_astronomical
function: kepler_orbits_pericenter_radius
Description: Return the pericenter radius for this orbit.
Code lines: 17
Contained by: module kepler_orbits

subroutine: kepler_orbits_pericenter_radius_set
Description: Sets the pericenter radius to the specified value.
Code lines: 13
Contained by: module kepler_orbits

subroutine: kepler_orbits_propagate
Description: Propagate an orbit along its path.
Code lines: 32
Contained by: module kepler_orbits
Modules used: galacticus_error numerical_constants_physical

function: kepler_orbits_radius
Description: Return the radius for this orbit.
Code lines: 12
Contained by: module kepler_orbits
Modules used: galacticus_error

subroutine: kepler_orbits_radius_set
Description: Sets the radius to the specified value.
Code lines: 13
Contained by: module kepler_orbits

subroutine: kepler_orbits_read_raw
Description: Read a keplerOrbit object in binary.
Code lines: 18
Contained by: module kepler_orbits

subroutine: kepler_orbits_reset
Description: Reset an orbit to a null state.
Code lines: 20
Contained by: module kepler_orbits

function: kepler_orbits_semi_major_axis
Description: Return the semi-major axis for this orbit.
Code lines: 18
Contained by: module kepler_orbits

subroutine: kepler_orbits_semi_major_axis_set
Description: Sets the semi-major axis to the specified value.
Code lines: 13
Contained by: module kepler_orbits
function: kepler_orbits_specific_reduced_mass
Description: Return the specific reduced mass for this orbit.
Code lines: 12
Contained by: module kepler_orbits
Modules used: galacticus_error

function: kepler_orbits_velocity_radial
Description: Return the radial velocity for this orbit.
Code lines: 22
Contained by: module kepler_orbits
Modules used: galacticus_error

subroutine: kepler_orbits_velocity_radial_set
Description: Sets the radial velocity to the specified value.
Code lines: 13
Contained by: module kepler_orbits

function: kepler_orbits_velocity_scale
Description: Return the velocity scale for the orbit.
Code lines: 15
Contained by: module kepler_orbits
Modules used: galacticus_error numerical_constants_physical

function: kepler_orbits_velocity_tangential
Description: Return the tangential velocity for this orbit.
Code lines: 27
Contained by: module kepler_orbits
Modules used: galacticus_error

subroutine: kepler_orbits_velocity_tangential_set
Description: Sets the tangential velocity to the specified value.
Code lines: 13
Contained by: module kepler_orbits

type: keplerorbit
Description: The structure used for describing orbits in GALACTICUS. This object will automatically convert from one set of orbital parameters to another where possible. The orbitting bodies (a satellite orbitting around its host) are treated as point masses, and the usual "reduced mass" framework is used, such that radii and velocities are measured relative to a stationary host. Energy and angular momentum are defined per unit satellite mass (not per unit reduced mass). Note that not all interconversions between elements are implemented. The object works by attempting to get the radial and tangential velocities and the radius. If it can obtain these, any other parameter can be computed. Getting these three parameters relies on having known conversions from other possible combinations of parameters.
Code lines: 273
Contained by: module kepler_orbits

file: objects.mass_distributions.F90
19. Source Code Documentation

**Description:** Contains a module which implements the mass distribution class.

**Code lines:** 1323

**module:** mass_distributions

**Description:** Implements the mass distribution class.

**Code lines:** 1303

**Contained by:** file objects.mass_distributions.F90

**Modules used:** fgsl

**Used by:** subroutine galacticus_state_retrieve subroutine galacticus_state_store module hot_halo_mass_distributions function node_component_spheroid_standard_half_mass_radius subroutine node_component_hot_halo_cold_mode_thread_initialize module node_component_hot_halo_cold_mode_structure_tasks module node_component_spheroid_standard_data

**function:** mass_distribution_beta_profile_density

**Description:** Return the density at the specified coordinates in a \( \beta \)-profile mass distribution.

**Code lines:** 15

**Contained by:** module mass_distributions

**Modules used:** coordinates

**function:** mass_distribution_beta_profile_density_gradient_radial

**Description:** Return the density at the specified coordinates in a \( \beta \)-profile mass distribution.

**Code lines:** 24

**Contained by:** module mass_distributions

**Modules used:** coordinates

**function:** mass_distribution_beta_profile_density_radial_moment

**Description:** Computes radial moments of the density in a \( \beta \)-profile mass distribution.

**Code lines:** 26

**Contained by:** module mass_distributions

**Modules used:** hypergeometric_functions

**subroutine:** mass_distribution_beta_profile_initialize

**Description:** Initialize a \( \beta \)-profile mass distribution.

**Code lines:** 68

**Contained by:** module mass_distributions

**Modules used:** galacticus_display galacticus_error hypergeometric_functions numerical_constants_math

**function:** mass_distribution_beta_profile_mass_enc_by_sphere

**Description:** Computes the mass enclosed within a sphere of given radius for \( \beta \)-profile mass distributions.

Result computed using Wolfram Alpha.

**Code lines:** 13

**Contained by:** module mass_distributions

**Modules used:** hypergeometric_functions numerical_constants_math
function: mass_distribution_beta_profile_potential
Description:  Return the potential at the specified coordinates in a β-profile mass distribution. Calculated using Wolfram Alpha.
Code lines: 36
Contained by: module mass_distributions
Modules used: coordinates hypergeometric_functions numerical_comparison numerical_constants_math numerical_constants_physical

function: mass_distribution_create
Description:  Create a mass distribution given the name.
Code lines: 19
Contained by: module mass_distributions
Modules used: galacticus_error

function: mass_distribution_density_gradient_radial_null
Description:  Aborts on attempts to get density of mass distributions with no density defined.
Code lines: 11
Contained by: module mass_distributions
Modules used: coordinates galacticus_error

function: mass_distribution_density_null
Description:  Aborts on attempts to get density of mass distributions with no density defined.
Code lines: 10
Contained by: module mass_distributions
Modules used: coordinates galacticus_error

function: mass_distribution_density_radial_moment_null
Description:  Aborts on attempts to get radial density moment of mass distributions with no density defined.
Code lines: 11
Contained by: module mass_distributions
Modules used: galacticus_error

function: mass_distribution_half_mass_radius_spherical
Description:  Aborts on attempts to get half-mass radius in spherical mass distributions.
Code lines: 8
Contained by: module mass_distributions
Modules used: galacticus_error

function: mass_distribution_hernquist_density
Description:  Return the density at the specified coordinates in a Hernquist mass distribution.
Code lines: 15
Contained by: module mass_distributions
Modules used: coordinates

function: mass_distribution_hernquist_density_radial_moment
Description:  Returns a radial density moment for the Hernquist mass distribution.
19. Source Code Documentation

**function:** mass_distribution_hernquist_half_mass_radius

*Description:* Return the half-mass radius of a Hernquist mass distribution.

*Code lines:* 8

**module:** mass_distributions

**Modules used:**
- galacticus_error
- numerical_comparison
- numerical_constants_math

**subroutine:** mass_distribution_hernquist_initialize

*Description:* Initialize a Hernquist mass distribution.

*Code lines:* 47

**module:** mass_distributions

**Modules used:**
- galacticus_error
- numerical_comparison
- numerical_constants_math

**function:** mass_distribution_hernquist_mass_enc_by_sphere

*Description:* Computes the mass enclosed within a sphere of given radius for Hernquist mass distributions.

*Code lines:* 11

**module:** mass_distributions

**Modules used:**
- numerical_constants_math

**function:** mass_distribution_hernquist_potential

*Description:* Return the potential at the specified coordinates in a Hernquist mass distribution.

*Code lines:* 15

**module:** mass_distributions

**Modules used:**
- coordinates
- numerical_constants_physical

**function:** mass_distribution_is_dimensionless

*Description:* Return true if self is a dimensionless mass distribution.

*Code lines:* 7

**module:** mass_distributions

**function:** mass_distribution_mass_enc_by_sphere_null

*Description:* Aborts on attempts to get the mass enclosed by a sphere for mass distributions with no density defined.

*Code lines:* 9

**module:** mass_distributions

**Modules used:**
- galacticus_error

**function:** mass_distribution_mass_enc_by_sphere_spherical

*Description:* Computes the mass enclosed within a sphere of given radius for spherically-symmetric mass distributions using numerical integration.

*Code lines:* 19

**module:** mass_distributions

**Modules used:**
- iso_c_binding
- numerical_constants_math
- numerical_integration
function: mass_distribution_mass_enc_by_sphere_spherical_integrand
Description: Enclosed mass integrand for spherical mass distributions.
Code lines: 13
Contained by: module mass_distributions
Modules used: coordinates iso_c_binding

function: mass_distribution_nfw_density
Description: Return the density at the specified coordinates in an NFW mass distribution.
Code lines: 15
Contained by: module mass_distributions
Modules used: coordinates

subroutine: mass_distribution_nfw_initialize
Description: Initialize an NFW mass distribution.
Code lines: 34
Contained by: module mass_distributions
Modules used: galacticus_error numerical_constants_math

function: mass_distribution_potential_null
Description: Aborts on attempts to get potential of mass distributions with no density defined.
Code lines: 10
Contained by: module mass_distributions
Modules used: galacticus_error

function: mass_distribution_sersic_density
Description: Return the density at the specified coordinates in a Sérsic mass distribution.
Code lines: 19
Contained by: module mass_distributions
Modules used: coordinates numerical_interpolation

function: mass_distribution_sersic_density_radial_moment
Description: Returns a radial density moment for the Sérsic mass distribution.
Code lines: 40
Contained by: module mass_distributions
Modules used: galacticus_error numerical_comparison numerical_constants_math

function: mass_distribution_sersic_half_mass_radius
Description: Return the half-mass radius of a Sérsic mass distribution.
Code lines: 9
Contained by: module mass_distributions

function: mass_distribution_sersic_half_mass_radius_projected
Description: Return the half-mass radius in projection of a Sérsic mass distribution.
Code lines: 9
Contained by: module mass_distributions
### subroutine: mass_distribution_sersic_initialize

*Description:* Initialize a Sérsic mass distribution.

*Code lines:* 53

*Contained by:* module *mass_distributions*

*Modules used:*
- *galacticus_error*
- *numerical_comparison*
- *numerical_constants_math*

### function: mass_distribution_sersic_mass_enc_by_sphere

*Description:* Computes the mass enclosed within a sphere of given radius for Sérsic mass distributions.

*Code lines:* 23

*Contained by:* module *mass_distributions*

*Modules used:*
- *numerical_constants_math*
- *numerical_interpolation*

### function: mass_distribution_sersic_potential

*Description:* Return the potential at the specified coordinates in a Sérsic mass distribution.

*Code lines:* 25

*Contained by:* module *mass_distributions*

*Modules used:*
- *coordinates*
- *numerical_constants_physical*
- *numerical_interpolation*

### function: mass_distribution_symmetry_cylindrical

*Description:* Returns symmetry label for mass distributions with cylindrical symmetry.

*Code lines:* 6

*Contained by:* module *mass_distributions*

### function: mass_distribution_symmetry_none

*Description:* Returns symmetry label for mass distributions with no symmetry.

*Code lines:* 6

*Contained by:* module *mass_distributions*

### function: mass_distribution_symmetry_spherical

*Description:* Returns symmetry label for mass distributions with spherical symmetry.

*Code lines:* 6

*Contained by:* module *mass_distributions*

### type: massdistribution

*Description:* The basic mass distribution class. Has no symmetry and will abort on inquiries.

*Code lines:* 56

*Contained by:* module *mass_distributions*

### type: massdistributionbeta_profile

*Description:* The β-profile: \( \rho(r) = \rho_0 / [1 + (r/r_{core})^2]^{3\beta/2} \)

*Code lines:* 19

*Contained by:* module *mass_distributions*

### type: massdistributioncylindrical

*Description:* A cylindrical mass distribution class.

*Code lines:* 4
19.1. Program units

**type: massdistributionhernquist**

*Description:* The Hernquist [Hernquist, 1990] density profile.
*Code lines:* 19
*Contained by:* module `mass_distributions`

**type: massdistributionnfw**

*Description:* The NFW [Navarro et al., 1996] density profile.
*Code lines:* 15
*Contained by:* module `mass_distributions`

**type: massdistributionsersic**

*Description:* The Sérsic density profile.
*Code lines:* 26
*Contained by:* module `mass_distributions`

**type: massdistributionspherical**

*Description:* A spherical mass distribution class.
*Code lines:* 15
*Contained by:* module `mass_distributions`

**function: sersic_abel_integrand**

*Description:* The integrand in the Abel integral used to invert the Sérsic profile to get the corresponding 3-D profile.
*Code lines:* 15
*Contained by:* module `mass_distributions`
*Modules used:* `iso_c_binding`, `numerical_constants_math`

**function: sersic_coefficient_root**

*Code lines:* 6
*Contained by:* module `mass_distributions`
*Modules used:* `gamma_functions`

**subroutine: sersic_profile_tabulate**

*Description:* Tabulate the density and enclosed mass in a dimensionless Sérsic profile.
*Code lines:* 118
*Contained by:* module `mass_distributions`
*Modules used:* `fgsl`, `iso_c_binding`, `memory_management`, `numerical_constants_math`, `numerical_integration`, `numerical_interpolation`, `numerical_ranges`, `root_finder`

**subroutine: sersic_profile_tabulate_state_retrieve**

*Description:* Retrieve the history state from the file.
*Code lines:* 10
*Contained by:* module `mass_distributions`
*Modules used:* `fgsl`
subroutine: sersic_profile_tabulate_state_store
Description: Write the history state to file.
Code lines: 9
Contained by: module mass_distributions
Modules used: fgsl

file: objects.merger_tree_data.F90
Description: Contains a module which implements an object to store merger tree data for processing into GALACTICUS's preferred file format.
Code lines: 2017

module: merger_tree_data_structure
Description: Implements an object to store merger tree data for processing into GALACTICUS's preferred file format.
Code lines: 1995
Contained by: file objects.merger_tree_data.F90
Modules used: iso_c_binding iso_varying_string kind_numbers

subroutine: merger_tree_data_construct_particle_indices
Description: If we have most-bound particle indices and particle data has been read, construct arrays giving position of particle data for each node.
Code lines: 40
Contained by: module merger_tree_data_structure
Modules used: galacticus_error memory_management

subroutine: merger_tree_data_set_subhalo_masses
Description: Set the masses of any subhalos (which have zero mass by default) based on particle count.
Code lines: 12
Contained by: module merger_tree_data_structure
Modules used: galacticus_display

subroutine: merger_tree_data_structure_add_metadata
Description: Add a metadata.
Code lines: 62
Contained by: module merger_tree_data_structure
Modules used: galacticus_error memory_management

subroutine: merger_tree_data_structure_add_metadata_double
Description: Add a double metadata.
Code lines: 12
Contained by: module merger_tree_data_structure
subroutine: merger_tree_data_structure_add_metadata_integer
  Description: Add an integer metadatum.
  Code lines: 10
  Contained by: module merger_tree_data_structure

subroutine: merger_tree_data_structure_add_metadata_text
  Description: Add a double metadatum.
  Code lines: 10
  Contained by: module merger_tree_data_structure

subroutine: merger_tree_data_structure_export
  Description: Output a set of merger trees to an HDF5 file.
  Code lines: 27
  Contained by: module merger_tree_data_structure
  Modules used: galacticus_error hdf5 string_handling

subroutine: merger_tree_data_structure_export_galacticus
  Description: Output a set of merger trees to a Galacticus-format HDF5 file.
  Code lines: 243
  Contained by: module merger_tree_data_structure
  Modules used: hdf5 io_hdf5 iso_c_binding memory_management string_handling

subroutine: merger_tree_data_structure_export_irate
  Description: Output a set of merger trees to an IRATE-format HDF5 file.
  Code lines: 271
  Contained by: module merger_tree_data_structure
  Modules used: array_utilities galacticus_error hdf5 io_hdf5 memory_management

subroutine: merger_tree_data_structure_make_references
  Description: Specify whether or not to make merger tree dataset references.
  Code lines: 8
  Contained by: module merger_tree_data_structure

subroutine: merger_tree_data_structure_read_ascii
  Description: Read in merger tree data from an ASCII file.
  Code lines: 330
  Contained by: module merger_tree_data_structure
  Modules used: file_utilities galacticus_display galacticus_error memory_management string_handling

subroutine: merger_tree_data_structure_read_particles_ascii
Description: Read in particle data from an ASCII file.
Code lines: 130
Contained by: module merger_tree_data_structure
Modules used: file_utilities galacticus_error memory_management string_handling

subroutine: merger_tree_data_structure_reset
Description: Reset a merger tree data object.
Code lines: 64
Contained by: module merger_tree_data_structure
Modules used: memory_management

subroutine: merger_tree_data_structure_set_includes_hubble_flow
Description: Set the particle mass used in the trees.
Code lines: 10
Contained by: module merger_tree_data_structure

subroutine: merger_tree_data_structure_set_includes_subhalo_masses
Description: Set the particle mass used in the trees.
Code lines: 10
Contained by: module merger_tree_data_structure

subroutine: merger_tree_data_structure_set_is_periodic
Description: Set whether or not positions are periodic.
Code lines: 10
Contained by: module merger_tree_data_structure

subroutine: merger_tree_data_structure_set_node_count
Description: Set the total number of nodes in merger trees.
Code lines: 10
Contained by: module merger_tree_data_structure

subroutine: merger_tree_data_structure_set_particle_count
Description: Set the total number of particles in merger trees.
Code lines: 10
Contained by: module merger_tree_data_structure

subroutine: merger_tree_data_structure_set_particle_mass
Description: Set the particle mass used in the trees.
Code lines: 10
Contained by: module merger_tree_data_structure

subroutine: merger_tree_data_structure_set_particle_property_column
Description: Set column mapping from the input file.
Code lines: 24
Contained by: module merger_tree_data_structure
Modules used: memory_management
19.1. Program units

**subroutine: merger_tree_data_structure_set_property_column**
- **Description:** Set column mapping from the input file.
- **Code lines:** 24
- **Contained by:** module `merger_tree_data_structure`
- **Modules used:** `memory_management`

**subroutine: merger_tree_data_structure_set_property_double**
- **Description:** Set a property in the merger trees.
- **Code lines:** 63
- **Contained by:** module `merger_tree_data_structure`
- **Modules used:** `galacticus_error` `memory_management`

**subroutine: merger_tree_data_structure_set_property_integer8**
- **Description:** Set a property in the merger trees.
- **Code lines:** 44
- **Contained by:** module `merger_tree_data_structure`
- **Modules used:** `galacticus_error` `memory_management`

**subroutine: merger_tree_data_structure_set_self_contained**
- **Description:** Set the particle mass used in the trees.
- **Code lines:** 10
- **Contained by:** module `merger_tree_data_structure`

**subroutine: merger_tree_data_structure_set_self_hosting_halo_id**
- **Description:** Set the host ID in case of self-hosting halos. Default is host ID = node ID.
- **Code lines:** 10
- **Contained by:** module `merger_tree_data_structure`

**subroutine: merger_tree_data_structure_set_tree_count**
- **Description:** Set the total number of trees in merger trees.
- **Code lines:** 10
- **Contained by:** module `merger_tree_data_structure`

**subroutine: merger_tree_data_structure_set_tree_indices**
- **Description:** Set the merger tree index arrays.
- **Code lines:** 41
- **Contained by:** module `merger_tree_data_structure`
- **Modules used:** `memory_management`

**subroutine: merger_tree_data_structure_set_units**
- **Description:** Set the units system.
- **Code lines:** 43
- **Contained by:** module `merger_tree_data_structure`
- **Modules used:** `galacticus_error`

**subroutine: merger_tree_data_validate_trees**
- **Description:** Validate the merger trees.
- **Code lines:** 11
19. Source Code Documentation

**Contained by:**  module **merger_tree_data_structure**
**Modules used:**  **galacticus_error**

**type:**  **mergertreedata**
**Description:**  A structure that holds raw merger tree data.
**Code lines:**  161
**Contained by:**  module **merger_tree_data_structure**

**subroutine:**  **store_unit_attributes_galacticus**
**Description:**  Store attributes describing the unit system.
**Code lines:**  13
**Contained by:**  module **merger_tree_data_structure**
**Modules used:**  **io_hdf5**

**subroutine:**  **store_unit_attributes_irate**
**Description:**  Store unit attributes in IRATE format files.
**Code lines:**  39
**Contained by:**  module **merger_tree_data_structure**
**Modules used:**  **io_hdf5**  **numerical_constants_prefixes**

**type:**  **treemetadata**
**Description:**  Structure that holds metadata for the trees.
**Code lines:**  10
**Contained by:**  module **merger_tree_data_structure**

**type:**  **unitsmetadata**
**Description:**  A structure that holds metadata on units used.
**Code lines:**  6
**Contained by:**  module **merger_tree_data_structure**

**file:**  **objects.merger_trees.dump.F90**
**Description:**  Contains a module which implements dumping of the structure of a merger tree to a file for plotting with DOT.
**Code lines:**  216

**module:**  **merger_trees_dump**
**Description:**  Implements dumping of the structure of a merger tree to a file for plotting with DOT.
**Code lines:**  196
**Contained by:**  file **objects.merger_trees.dump.F90**
**Modules used:**  **iso_c_binding**  **kind_numbers**
**Used by:**  subroutine **merger_tree_structure_dump**  subroutine **merger_tree_evolve_to**
subroutine **merger_tree_regrid_time**

**subroutine:**  **merger_tree_dump**
| **Description:** | Dumps the tree structure to a file in a format suitable for processing with **dot**. Nodes are shown as circles if isolated or rectangles if satellites. Isolated nodes are connected to their descendent halo, while satellites are connected (by red lines) to their host halo. Optionally, a list of node indices to highlight can be specified. |
| **Code lines:** | 180 |
| **Contained by:** | module **merger_trees_dump** |
| **Modules used:** | **galacticus_nodes**  
**iso_varying_string** |
file: objects.merger_trees.information_content.F90
Description: Contains a module which computes the cladistic information content (CIC) of a merger tree.
Code lines: 69

module: merger_trees_information_content
Description: Implements computation of the cladistic information content (CIC) of a merger tree.
Code lines: 49
Contained by: file objects.merger_trees.information_content.F90

function: merger_tree_cladistic_information_content
Description: Compute the cladistic information content (CIC; Thorley et al. 1998) of a merger tree in bits.
Code lines: 39
Contained by: module merger_trees_information_content
Modules used: factorials galacticus_nodes

file: objects.nodes.F90
Description: Contains a module which implements an object hierarchy for nodes in merger trees and all of their constituent physical components.
Code lines: 57195

module: galacticus_nodes

Code lines: 57174
Contained by: file objects.nodes.F90
Modules used: abundances_structure chemical_abundances_structure galacticus_error histories io_hdf5 iso_varying_string kepler_orbits memory_management numerical_constants_astronomical stellar_luminosities_structure tensors

Used by: module accretion_halos function coldmodeaccretedmass function coldmodeaccretedmasschemicals function coldmodeaccretedmassmetals function coldmodeaccretionrate function coldmodeaccretionratechemicals function coldmodeaccretionratemetal function coldmodeaccretionratetotal function coldmodechemicalmasses function coldmodecoldmodefraction function coldmodefailedaccretedmass function coldmodefailedaccretionrate function coldmodefailedaccretionratechemicals function coldmodefailedaccretionratetotal function coldmodefailedaccretedmass function coldmodefailedaccretedmasschemicals function coldmodefailedaccretedmassmetals function coldmodefailedaccretionrate function coldmodefailedaccretionratechemicals function coldmodefailedaccretionratemetal function coldmodefailedaccretionratetotal function coldmodefailedaccretedmass function coldmodefailedaccretedmasschemicals function coldmodefailedaccretedmassmetals function coldmodefailedaccretionrate function coldmodefailedaccretionratechemicals function coldmodefailedaccretionratemetal function coldmodefailedaccretionratetotal function coldmodefailedaccretedmass
module cooling_radii_isothermal
function cooling_rate_cole2000
module cooling_rates_white_frenk
module cooling_rates_simple
function cooling_rate_zero
function freefall_radius_dark_matter_halo
module cooling_freefall_times_available
module cooling_infall_radii
function infall_radius_growth_rate_cooling_freefall
function infall_radius_growth_rate_cooling_radius
subroutine cooling_specific_am_constant_rotation_reset
subroutine cooling_specific_am_mean_initialize
module cooling_times_available
function cooling_time_available_wf
function dark_matter_halo_formation_time
module dark_matter_halo_mass_accretion_histories
module dark_matter_halo_mass_accretionHistories_wechsler2002
module dark_matter_halos_mass_loss_rates
function dark_matter_halos_mass_loss_rate_vandenbosch
function dark_matter_haloAngularMomentum
subroutine dark_matter_halo_spins_initialize
module halo_spin_distributions
function halo_spin_distribution_lognormal
function isothermal_circular_velocity
function isothermal_density
function isothermal_energy
function isothermal_free_fall_radius
module cooling_radii_simple
module cooling_rates
subroutine cooling_rate_modifier_cut_off
module cooling_rates_simple_scaling
module freefall_radii
function freefall_radius_growth_rate_dark_matter_halo
module freefall_times_available_halo_formation
function infall_radius_cooling_freefall
function infall_radius_cooling_radius
module cooling_specific_angular_momenta
function cooling_specific_angular_momentum_constant_rotation
function cooling_specific_angular_momentum_mean
function cooling_time_available_increase_rate_wf
module cooling_times_available_halo_formation
module dark_matter_halo_mass_accretion_histories
module dark_matter_halo_mass_accretionHistories_zhao2009
module dark_matter_halos_mass_loss_rate_null
module dark_matter_halo_scales
function dark_matter_halo_angular_momentum_growth_rate
function halo_spin_distribution_bett2007
function halo_spin_distribution_delta
module dark_matter_profiles
function isothermal_circular_velocity_maximum
function isothermal_enclosed_mass
function isothermal_energy_growth_rate
function isothermal_free_fall_radius_increase_rate
function isothermalspace
function isothermalradiusfromspecificangularmomentum
function isothermalcircularvelocity
function isothermdensity
function isothermalenergy
function isothermalfreefallradius
function isothermalspace
function isothermalradiusfromspecificangularmomentum
function isothermalrotationnormalization
module dark_matter_profiles_specificangularmomentum
function dark_matter_profile_scale
module dark_matter_profiles_shapes
function dark_matter_profile_shape_gao2008
module dark_matter_profile_structure_tasks
subroutine event_halo_formation
subroutine node_promotion_index_shift
subroutine bar_instability_timescale_eln
subroutine bar_instability_timescale_eln_tidal
module galactic_dynamics_bar_instabilities
subroutine bar_instability_timescale_eln_tidal
module galactic_dynamics_bar_instabilities_null
function component_density
module galactic_structure_density
module galactic_structure_potentials
module galactic_structure_initial_radii
module galactic_structure_initial_radii_static
module galactic_structure_rotation_curves
module galactic_structure_surface_densities
subroutine galacticus_calculations_reset
module galacticus_output_analyses_mass_dpndnt_sz_dstrbtins
module galacticus_output_analyses_mass_functions
subroutine count_properties
subroutine galacticus_merger_tree_output
subroutine event_halo_formation
subroutine node_promotion_index_shift
subroutine bar_instability_timescale_eln
subroutine bar_instability_timescale_eln_tidal
module galactic_dynamics_bar_instabilities
subroutine bar_instability_timescale_eln_tidal
module galactic_dynamics_bar_instabilities_null
function component_density
module galactic_structure_density
module galactic_structure_potentials
module galactic_structure_initial_radii
module galactic_structure_initial_radii_static
module galactic_structure_rotation_curves
module galactic_structure_surface_densities
subroutine galacticus_calculations_reset
module galacticus_output_analyses_mass_dpndnt_sz_dstrbtins
module galacticus_output_analyses_mass_functions
subroutine count_properties
subroutine galacticus_merger_tree_output
module galacticus_output_trees_density_contrasts
function galacticus_merger_tree_output_filter
subroutine galacticus_output_tree_lightcone
subroutine galacticus_output_tree_lightcone_property_count
subroutine galacticus_merger_tree_output_filter_luminosity
subroutine galacticus_merger_tree_output_stellar_mass
subroutine galacticus_merger_tree_output_filter_stellar_mass
module galacticus_output_trees_descendents
subroutine galacticus_output_tree_descendents
subroutine galacticus_output_tree_descendents_property_count
module galacticus_output_tree_half_light_properties
subroutine galacticus_extra_output_halo_fourier_profile
subroutine galacticus_output_halo_model
subroutine galacticus_output_halo_model_names
subroutine galacticus_output_halo_model_property_count
module galacticus_output_trees_final_descendents
subroutine galacticus_output_tree_final_descendents
subroutine galacticus_output_tree_final_descendents_property_count
module galacticus_output_trees_links
subroutine galacticus_output_tree_main_branch
subroutine galacticus_output_tree_main_branch_names
subroutine galacticus_output_tree_main_branch_property_count
module galacticus_output_tree_mass_profiles
subroutine galacticus_output_most_massive_progenitor
subroutine galacticus_output_most_massive_progenitor_names
subroutine galacticus_output_most_massive_progenitor_property_count
module galacticus_output_trees_redshifts
subroutine galacticus_output_tree_rotation_curve
subroutine galacticus_output_tree_rotation_curve_initialize
subroutine galacticus_output_tree_rotation_curve_property_count
subroutine galacticus_output_tree_satellite_extremum
subroutine galacticus_output_tree_satellite_extremum_names
subroutine galacticus_output_tree_satellite_host
subroutine galacticus_output_tree_satellite_host_names
subroutine galacticus_output_tree_satellite_host_property_count
subroutine galacticus_output_tree_satellite_status
subroutine galacticus_output_tree_satellite_status_initialize
subroutine galacticus_output_tree_satellite_status_property_count
module galacticus_output_star_formation_histories

subroutine star_formation_history_create_metallicity_split
subroutine star_formation_history_record_metallicity_split
subroutine star_formation_history_output_metallicity_split
module galacticus_output_trees_tree_indices
subroutine galacticus_output_tree_virial
subroutine galacticus_output_tree_virial_property_count
subroutine get_tree
module hot_halo_cold_mode_density_cored_isothermal_core_radii
module hot_halo_mass_distributions
function hot_halo_ram_pressure_force
function hot_halo_ram_pressure_force_null_get
module hot_halo_ram_pressure_stripping_font2008
function hot_halo_ram_pressure_timescale_halo_dynetime
function hot_halo_ram_pressure_stripping_virial_radius
function hot_halo_temperature_logarithmic_slope_virial_get
subroutine intergalactic_medium_state_internal_initialize
subroutine merger_tree_conditional_mass_function
subroutine merger_tree_build_do_cole2000
function merger_tree_build_mass_resolution
function merger_tree_build_mass_resolution_scaled
function node_lookup
module merger_tree_read_importers
subroutine merger_tree_state_restore
subroutine merger_tree_state_walk_tree
subroutine merger_tree_dump_evolution
subroutine star_formation_history_output_null
module galacticus_output_trees_velocity_dispersion
subroutine galacticus_output_tree_virial_names
function galacticus_task_evolve_tree
subroutine halo_mass_function_compute
function hot_halo_cold_mode_density_ciso_corer_vfrac
module hot_halo_mass_distributions_core_radii
module hot_halo_mass_distributions
function hot_halo_ram_pressure_force_font2008
function hot_halo_ram_pressure_stripping_radius
function hot_halo_ram_pressure_stripping_timescale
function hot_halo_ram_pressure_timescale_ram_pressure_accel
module hot_halo_temperature_profile
function hot_halo_temperature_virial_get
function intergalactic_medium_state_internal_update
module merger_tree_construction
module merger_tree_build
function merger_tree_build_mass_resolution_fixed
subroutine merger_tree_construct_fully_specified
module merger_tree_read
subroutine merger_tree_smooth_accretion_do
subroutine merger_tree_state_store
function pointed_at_node
subroutine merger_tree_structure_dump
<table>
<thead>
<tr>
<th>Module</th>
<th>Subroutine</th>
</tr>
</thead>
<tbody>
<tr>
<td>merger_trees_evolve</td>
<td>module merger_trees_evolve_node</td>
</tr>
<tr>
<td>merge_tree_timesteps_history</td>
<td>subroutine merger_tree_record_evolution_output</td>
</tr>
<tr>
<td></td>
<td>subroutine satellite_merger_process</td>
</tr>
<tr>
<td></td>
<td>subroutine merger_tree_monotonic_mass_growth</td>
</tr>
<tr>
<td></td>
<td>subroutine merger_tree_non_essential_branches</td>
</tr>
<tr>
<td></td>
<td>subroutine merger_trees_timestep_simple</td>
</tr>
<tr>
<td></td>
<td>subroutine merger_tree_mass_accretion_history_output</td>
</tr>
<tr>
<td></td>
<td>subroutine merger_tree_structure_output</td>
</tr>
<tr>
<td></td>
<td>subroutine merger_tree_prune_branches</td>
</tr>
<tr>
<td></td>
<td>subroutine merger_tree_prune_hierarchy</td>
</tr>
<tr>
<td></td>
<td>subroutine merger_trees_render_dump</td>
</tr>
<tr>
<td></td>
<td>subroutine merger_tree_dump</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_agestatistics_diskintegratedsfr</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_agestatistics_disktimeweightedintegratedsfr</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_agestatistics_spheroidintegratedsfr</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_agestatistics_spheroidtimeweightedintegratedsfr</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_accretionrate</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_massmaximum</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_timelastisolated</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_blackhole_accretionrate</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_blackhole_massseed</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_blackhole_radialposition</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_blackhole_spin</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_blackhole_tripleinteractiontime</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_darkmatterprofile_scale</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_darkmatterprofile_shape</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_disk_angularmomentum</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_disk_massgas</td>
</tr>
<tr>
<td></td>
<td>function cnode_component_disk_massstellar</td>
</tr>
</tbody>
</table>
function cnode_component_disk_-
starformationrate
function cnode_component_- 
formationtime_formationtime
function cnode_component_hothalo_- 
angularmomentum
function cnode_component_hothalo_mass

function cnode_component_hothalo_- 
outerradius
function cnode_component_hothalo_- 
outflowedmass
function cnode_component_hothalo_- 
trackedoutflowmass
function cnode_component_interoutput_- 
diskstarformationrate
function cnode_component_- 
massflowstatistics_cooledmass

function cnode_component_- 
mergingstatistics_nodeformationtime
function cnode_component_nbody_- 
velocitydispersion
function cnode_component_satellite_- 
boundmass
function cnode_component_satellite_- 
tidalheatingnormalized
function cnode_component_spheroid_- 
angularmomentum
function cnode_component_spheroid_- 
massgas
function cnode_component_spheroid_- 
radius
function cnode_component_spheroid_- 
velocity
function cnode_component_spin_- 
spingrowthrate
subroutine node_component_age_- 
statistics_standard_rate_compute
module node_component_basic_standard

module node_component_black_hole_- 
simple
function node_component_black_hole_- 
simple_rotation_curve
module node_component_black_hole_standard
function node_component_black_hole_standard_rotation_curve
module node_component_dark_matter_profile_scale
module node_component_dark_matter_profile_scale_shape
module node_component_disk_exponential
module node_component_disk_very_simple
subroutine node_component_dynamics_statistics_bars
module node_component_host_history_standard
module node_component_hot_halo_cold_mode
function node_component_hot_halo_cold_mode_density_task
function node_component_hot_halo_cold_mode_enclosed_mass_task
function node_component_hot_halo_cold_mode_rotation_curve_gradient_task
function node_component_hot_halo_cold_mode_rotation_curve_task
module node_component_hot_halo_outflow_tracking
module node_component_hot_halo_very_simple
module node_component_inter_output_standard
subroutine node_component_mass_flow_statistics_standard_extra_output
module node_component_mass_flow_statistics_standard
module node_component_merging_statistics_recent
module node_component_merging_statistics_standard
module node_component_nbody_standard
module node_component_position_preset
module node_component_satellite_orbiting
module node_component_satellite_preset
module node_component_satellite_very_simple
module node_component_spin_preset
module node_component_spin_random
subroutine radiation_set_time
module radiation_intergalactic_background
subroutine radiation_intergalactic_background_internalInitialize
subroutine radiation_set_null
function ram_pressure_stripping_mass_loss_rate_disk
function ram_pressure_stripping_mass_loss_rate_disk_null
function ram_pressure_stripping_mass_loss_rate_spheroid
function ram_pressure_stripping_mass_loss_rate_spheroid_null
function ram_pressure_stripping_mass_loss_rate_spheroid_simple
function satellite_dynamical_friction_acceleration
subroutine satellite_merging_mass_movementbaugh2005
subroutine satellite_merging_mass_movement_store
subroutine satellite_merging_mass_movement_very_simple
subroutine satellite_merging_remnant_size_covevington2008
subroutine satellite_merging_remnant_size_null
subroutine satellite_merging_remnant_progenitor_properties
subroutine satellite_merging_remnant_progenitor_properties_standard_init
module satellite_merging_timescales

function infinitetimeuntilmerging
function laceycole1993timeuntilmerging

function laceycole1993tormentimeuntilmerging
function presettimeuntilmerging
function wetzel2010timeuntilmerging
function laceycole1993timeuntilmerging

function laceycole1993tormentimeuntilmerging
function wetzel2010timeuntilmerging
function nulltimeuntilmerging
function virial_orbital_parameters_benson2005
function virial_orbital_parameters_wetzel2010
module satellite_orbits
function satellite_tidal_field

module ram_pressure_stripping_mass_loss_rate_spheroids_null
module ram_pressure_stripping_mass_loss_rate_spheroids_simple
function satellite_dynamical_friction_acceleration_chandrasekhar
function satellite_dynamical_friction_acceleration_null
subroutine satellite_merging_mass_movement
subroutine satellite_merging_mass_movement_simple
subroutine satellite_merging_mass_movement_very_simple
subroutine satellite_merging_remnant_size
subroutine satellite_merging_remnant_size_covevington2008
subroutine satellite_merging_remnant_progenitor_properties
subroutine satellite_merging_remnant_progenitor_properties_standard
function boylankolchin2008timeuntilmerging
function laceycole1993timeuntilmergingmassdependence
function nulltimeuntilmerging
subroutine satellite_move_to_new_host
module satellites_tidal_fields_null

function jiang2008defaultconstructor
function laceycole1993timeuntilmergingmassdependence
function nulltimeuntilmerging
function villalobos2013timeuntilmerging
function jiang2008defaultconstructor
function laceycole1993timeuntilmergingmassdependence
function nulltimeuntilmerging
function wetzel2010timeuntilmerging
function laceycole1993timeuntilmerging

function presettimeuntilmerging
module virial_orbits
function virial_orbitalParameters_fixed
subroutine satellite_move_to_new_host
module satellites_tidal_fields_null
function satellites_tidal_field_null_get
function satellites_tidal_fields_spherical_symmetry_get
function satellite_tidal_heating_rate_gnedin
function satellite_tidal_heating_rate_null
function satellite_tidal_stripping_rate
function satellite_tidal_stripping_rate_zentner2005
module star_formation_feedback_disks_creasey2012
module star_formation_feedback_disks_fixed
function star_formation_feedback_disk_outflow_rate_power_law
function star_formation_feedback_spheroid_outflow_rate_power_law
function star_formation_expulsive_feedback_disk_outflow_rate_null
module star_formation_feedback_expulsion_spheroids
function star_formation_expulsive_feedback_spheroid_outflow_rate_null
function star_formation_expulsive_feedback_spheroid_outflow_rate_sw
module star_formation_rate_surface_density_disks_br
module star_formation_rate_surface_density_disks
module star_formation_rate_surface_density_disks_kmt09
module star_formation_rate_surface_density_disks_ks
module star_formation_rate_surface_density_disks_exschmidt
module star_formation_timescales_spheroids
subroutine star_formation_timescale_disks_dynamical_time_initialize
module star_formation_timescale_disks_halo_scaling
module star_formation_timescales_spheroids
module stellar_population_properties
subroutine stellar_population_properties_history_create_instantaneous
subroutine stellar_population_properties_history_create_noninstantaneous
subroutine stellar_population_properties_rates_instantaneous
subroutine stellar_population_properties_rates_noninstantaneous
module satellites_tidal_fields_spherical_symmetry
function satellite_tidal_heating_rate
function satellite_tidal_heating_rate_null
module tidal_stripping_rate_zentner2005
function satellite_tidal_stripping_rate_null
module star_formation_feedback_disks_halo_scaling
module star_formation_feedback_spheroids
module star_formation_feedback_expulsion_disks
function star_formation_expulsive_feedback_disk_outflow_rate_sw
module star_formation_expulsive_feedback_spheroids
function star_formation_expulsive_feedback_spheroid_outflow_rate_null
module star_formation_rate_surface_density_disks
module star_formation_rate_surface_density_disks_kmt09
module star_formation_rate_surface_density_disks_exschmidt
module star_formation_timescales_spheroids
subroutine star_formation_timescale_disk_dynamical_time
function star_formation_timescale_disk_fixed
module star_formation_timescale_disks_integrated_sd
module stellar_population_properties
subroutine stellar_population_properties_history_create_instantaneous
subroutine stellar_population_properties_history_create_noninstantaneous
module dark_matter_halo_biases
function dark_matter_halo_bias_node_press_schechter
function dark_matter_halo_bias_node_tinker2010
program test_nfw96_concentration_dark_energy
program test_zhao2009_flat
program test_zhao2009_open
program test_nodes
function testfuncdouble0
program tests_tree_branch_destroy
module tidal_stripping_mass_loss_rate_disk
module tidal_stripping_mass_loss_rate_disks_null
module tidal_stripping_mass_loss_rate_disks_simple
function tidal_stripping_mass_loss_rate_disk_null
module tidal_stripping_mass_loss_rate_spheroids_null
module tidal_stripping_mass_loss_rate_spheroids_simple
function tidal_stripping_mass_loss_rate_spheroid
function tidal_stripping_mass_loss_rate_spheroid_null
function tidal_stripping_mass_loss_rate_spheroid_simple

function: agestatisticscountlinked
Description: Returns the number of ageStatistics components in self.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: agestatisticscreatebyinterrupt
Description: Create the ageStatistics component of self via an interrupt.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: agestatisticscreatelinked
Description: Create the ageStatistics component of self.
Code lines: 29
Contained by: module galacticus_nodes

Modules used: galacticus_display iso_varying_string string_handling

subroutine: agestatisticsdestroylinked
Description: Destroy the ageStatistics component of self.
Code lines: 13
Contained by: module galacticus_nodes

function: agestatisticsdiskintegratedsfr
Description: Returns the default value for the diskIntegratedSFR property for the ageStatistics component class.
Code lines: 12
19.1. Program units

**function: agestatisticsdiskintegratedsfrattributematch**
*Description:* Return a text list of component implementations in the *ageStatistics* class that have the desired attributes for the *diskIntegratedSFR* property
*Code lines:* 25
*Contained by:* module *galacticus_nodes*
*Modules used:* *iso_varying_string*

**function: agestatisticsdiskintegratedsfrisgettable**
*Description:* Returns true if the *diskIntegratedSFR* property is gettable for the *ageStatistics* component class.
*Code lines:* 7
*Contained by:* module *galacticus_nodes*

**subroutine: agestatisticsdiskintegratedsfrrate**
*Description:* Accept a rate set for the *diskIntegratedSFR* property of the *ageStatistics* component class. Trigger an interrupt to create the component.
*Code lines:* 15
*Contained by:* module *galacticus_nodes*
*Modules used:* *galacticus_error*

**function: agestatisticsdisktimeweightedintegratedsfr**
*Description:* Returns the default value for the *diskTimeWeightedIntegratedSFR* property for the *ageStatistics* component class.
*Code lines:* 12
*Contained by:* module *galacticus_nodes*

**function: agestatisticsdisktimeweightedintegratedsfrattributematch**
*Description:* Return a text list of component implementations in the *ageStatistics* class that have the desired attributes for the *diskTimeWeightedIntegratedSFR* property
*Code lines:* 25
*Contained by:* module *galacticus_nodes*
*Modules used:* *iso_varying_string*

**function: agestatisticsdisktimeweightedintegratedsfrisgettable**
*Description:* Returns true if the *diskTimeWeightedIntegratedSFR* property is gettable for the *ageStatistics* component class.
*Code lines:* 7
*Contained by:* module *galacticus_nodes*

**subroutine: agestatisticsdisktimeweightedintegratedsfrrate**
*Description:* Accept a rate set for the *diskTimeWeightedIntegratedSFR* property of the *ageStatistics* component class. Trigger an interrupt to create the component.
*Code lines:* 15
*Contained by:* module *galacticus_nodes*
*Modules used:* *galacticus_error*

**function: agestatisticsget**
Source Code Documentation

**Description:** Returns the `ageStatistics` component of `self`.

**Code lines:** 27

**Contained by:** module `galacticus_nodes`

**Modules used:** `galacticus_error`

**function: agestatisticsnullbindingdouble0inout**

**Description:** A null get function for rank 0 double precisions.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**function: agestatisticsnullbindinginteger0in**

**Description:** A null get function for rank 0 integers.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**subroutine: agestatisticsnullbindingratedouble0inout**

**Description:** A null rate function for rank 0 double precisions.

**Code lines:** 9

**Contained by:** module `galacticus_nodes`

**subroutine: agestatisticsnullbindingrateinteger0in**

**Description:** A null rate function for rank 0 integers.

**Code lines:** 9

**Contained by:** module `galacticus_nodes`

**subroutine: agestatisticsnullbindingsetdouble0inout**

**Description:** A null set function for rank 0 double precisions.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**subroutine: agestatisticsnullbindingsetinteger0in**

**Description:** A null set function for rank 0 integers.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**function: agestatisticsspheroidintegratedsfr**

**Description:** Returns the default value for the `spheroidIntegratedSFR` property for the `ageStatistics` component class.

**Code lines:** 12

**Contained by:** module `galacticus_nodes`

**function: agestatisticsspheroidintegratedsfrattributematch**

**Description:** Return a text list of component implementations in the `ageStatistics` class that have the desired attributes for the `spheroidIntegratedSFR` property

**Code lines:** 25

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function: agestatisticsspheroidintegratedsfrisgettable**
19.1. Program units

**Description:** Returns true if the `spheroidIntegratedSFR` property is gettable for the `ageStatistics` component class.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**subroutine:** `agestatisticsspheroidintegratedsfrrate`

**Description:** Accept a rate set for the `spheroidIntegratedSFR` property of the `ageStatistics` component class. Trigger an interrupt to create the component.

**Code lines:** 15

**Contained by:** module `galacticus_nodes`

**Modules used:** `galacticus_error`

**function:** `agestatisticsspheroidtimeweightedintegratedsfr`

**Description:** Returns the default value for the `spheroidTimeWeightedIntegratedSFR` property for the `ageStatistics` component class.

**Code lines:** 12

**Contained by:** module `galacticus_nodes`

**function:** `agestatisticsspheroidtimeweightedintegratedsfrrate`

**Description:** Accept a rate set for the `spheroidTimeWeightedIntegratedSFR` property of the `ageStatistics` component class. Trigger an interrupt to create the component.

**Code lines:** 15

**Contained by:** module `galacticus_nodes`

**Modules used:** `galacticus_error`

**function:** `agestatisticsspheroidtimeweightedintegratedsfrrate`

**Description:** Accept a rate set for the `spheroidTimeWeightedIntegratedSFR` property of the `ageStatistics` component class. Trigger an interrupt to create the component.

**Code lines:** 15

**Contained by:** module `galacticus_nodes`

**Modules used:** `galacticus_error`

**function:** `agestatisticsstandarddiskintegratedsfrcount`

**Description:** Return a count of the number of scalar properties in the `diskIntegratedSFR` property of the `AgeStatisticsStandard` component implementation.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**function:** `agestatisticsstandarddiskintegratedsfrget`

**Description:** Return the `diskIntegratedSFR` property of the `AgeStatisticsStandard` component implementation.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`
subroutine: agestatisticsstandarddiskintegratedsfrrate
Description: Accumulate to the diskIntegratedSFR property rate of change of the AgeStatisticsStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: agestatisticsstandarddiskintegratedsfrscale
Description: Set the diskIntegratedSFR property scale of the AgeStatisticsStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: agestatisticsstandarddiskintegratedsfrset
Description: Set the diskIntegratedSFR property of the AgeStatisticsStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: agestatisticsstandarddisktimeweightedintegratedsfrcount
Description: Return a count of the number of scalar properties in the diskTimeWeightedIntegratedSFR property of the ageStatisticsStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: agestatisticsstandarddisktimeweightedintegratedsfrget
Description: Return the diskTimeWeightedIntegratedSFR property of the AgeStatisticsStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: agestatisticsstandarddisktimeweightedintegratedsfrrate
Description: Accumulate to the diskTimeWeightedIntegratedSFR property rate of change of the AgeStatisticsStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: agestatisticsstandarddisktimeweightedintegratedsfrscale
Description: Set the diskTimeWeightedIntegratedSFR property scale of the AgeStatisticsStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: agestatisticsstandarddisktimeweightedintegratedsfrset
Description: Set the diskTimeWeightedIntegratedSFR property of the AgeStatisticsStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management
function: agestatisticsstandardspheroidintegratedsfrcount
Description: Return a count of the number of scalar properties in the spheroidIntegratedSFR property of the ageStatisticsStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: agestatisticsstandardspheroidintegratedsfrget
Description: Return the spheroidIntegratedSFR property of the AgeStatisticsStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: agestatisticsstandardspheroidintegratedsfrrate
Description: Accumulate to the spheroidIntegratedSFR property rate of change of the AgeStatisticsStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: agestatisticsstandardspheroidintegratedsfrscale
Description: Set the spheroidIntegratedSFR property scale of the AgeStatisticsStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: agestatisticsstandardspheroidintegratedsfrset
Description: Set the spheroidIntegratedSFR property of the AgeStatisticsStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: agestatisticsstandardspheroidtimeweightedintegratedsfrcount
Description: Return a count of the number of scalar properties in the spheroidTimeWeightedIntegratedSFR property of the ageStatisticsStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: agestatisticsstandardspheroidtimeweightedintegratedsfrget
Description: Return the spheroidTimeWeightedIntegratedSFR property of the AgeStatisticsStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: agestatisticsstandardspheroidtimeweightedintegratedsfrrate
Description: Accumulate to the spheroidTimeWeightedIntegratedSFR property rate of change of the AgeStatisticsStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes
19. Source Code Documentation

**subroutine**: agestatisticsstandardspheroidtimeweightedintegratedsfrscale

*Description:* Set the `spheroidTimeWeightedIntegratedSFR` property scale of the `AgeStatisticsStandard` component implementation.

*Code lines:* 8  
*Contained by:* module `galacticus_nodes`

**subroutine**: agestatisticsstandardspheroidtimeweightedintegratedsfrset

*Description:* Set the `spheroidTimeWeightedIntegratedSFR` property of the `AgeStatisticsStandard` component implementation.

*Code lines:* 11  
*Contained by:* module `galacticus_nodes`

*Modules used:* `memory_management`

**interface**: assignment(=)

*Code lines:* 2  
*Contained by:* module `galacticus_nodes`

**function**: basicaccretionrate

*Description:* Returns the default value for the `accretionRate` property for the `basic` component class.

*Code lines:* 8  
*Contained by:* module `galacticus_nodes`

**function**: basicaccretionrateattributematch

*Description:* Return a text list of component implementations in the `basic` class that have the desired attributes for the `accretionRate` property

*Code lines:* 27  
*Contained by:* module `galacticus_nodes`

*Modules used:* `iso_varying_string`

**function**: basicaccretionrateisgettable

*Description:* Returns true if the `accretionRate` property is gettable for the `basic` component class.

*Code lines:* 6  
*Contained by:* module `galacticus_nodes`

**function**: basiccountlinked

*Description:* Returns the number of `basic` components in `self`.

*Code lines:* 19  
*Contained by:* module `galacticus_nodes`

**subroutine**: basiccreatebyinterrupt

*Description:* Create the `basic` component of `self` via an interrupt.

*Code lines:* 8  
*Contained by:* module `galacticus_nodes`

**subroutine**: basiccreatelinked

*Description:* Create the `basic` component of `self`.

*Code lines:* 29  
*Contained by:* module `galacticus_nodes`
19.1. Program units

Modules used:  
galacticus_display  
iso_varying_string  
string_handling

subroutine: basicdestroylinked  
Description: Destroy the basic component of self.  
Code lines: 13  
Contained by: module galacticus_nodes

function: basicget  
Description: Returns the basic component of self.  
Code lines: 27  
Contained by: module galacticus_nodes  
Modules used: galacticus_error

function: basicmass  
Description: Returns the default value for the mass property for the basic component class.  
Code lines: 8  
Contained by: module galacticus_nodes

function: basicmassattributematch  
Description: Return a text list of component implementations in the basic class that have the desired attributes for the mass property  
Code lines: 36  
Contained by: module galacticus_nodes  
Modules used: iso_varying_string

function: basicmassisgettable  
Description: Returns true if the mass property is gettable for the basic component class.  
Code lines: 6  
Contained by: module galacticus_nodes

function: basicmassmaximum  
Description: Returns the default value for the massMaximum property for the basic component class.  
Code lines: 8  
Contained by: module galacticus_nodes

function: basicmassmaximumattributematch  
Description: Return a text list of component implementations in the basic class that have the desired attributes for the massMaximum property  
Code lines: 27  
Contained by: module galacticus_nodes  
Modules used: iso_varying_string

function: basicmassmaximumisgettable  
Description: Returns true if the massMaximum property is gettable for the basic component class.  
Code lines: 6  
Contained by: module galacticus_nodes

function: basicnonevolvingmassget  
Description: Return the mass property of the BasicNonEvolving component implementation.
subroutine: basicnonevolvingmassset
Description: Set the mass property of the BasicNonEvolving component implementation.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

function: basicnonevolvingtimecount
Description: Return a count of the number of scalar properties in the time property of the basicNonEvolving component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: basicnonevolvingtimeget
Description: Return the time property of the BasicNonEvolving component implementation.
Code lines: 8
Contained by: module galacticus_nodes

function: basicnonevolvingtimelastisolated
Description: Return the timeLastIsolated property of the Basic component class.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: basicnonevolvingtimelastisolatedset
Description: Set the timeLastIsolated property of the BasicNonEvolving component implementation.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: basicnonevolvingtimerate
Description: Accumulate to the time property rate of change of the BasicNonEvolving component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: basicnonevolvingtimescale
Description: Set the time property scale of the BasicNonEvolving component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: basicnonevolvingtimeset
Description: Set the time property of the BasicNonEvolving component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: basicnullbindingdouble0inout
19.1. Program units

**Description:** A null get function for rank 0 double precisions.
**Code lines:** 7
**Contained by:** module `galacticus_nodes`

**function:** `basicnullbindinginteger0in`
**Description:** A null get function for rank 0 integers.
**Code lines:** 7
**Contained by:** module `galacticus_nodes`

**subroutine:** `basicnullbindingratedouble0inout`
**Description:** A null rate function for rank 0 double precisions.
**Code lines:** 9
**Contained by:** module `galacticus_nodes`

**subroutine:** `basicnullbindingrateinteger0in`
**Description:** A null rate function for rank 0 integers.
**Code lines:** 9
**Contained by:** module `galacticus_nodes`

**subroutine:** `basicnullbindingsetdouble0inout`
**Description:** A null set function for rank 0 double precisions.
**Code lines:** 7
**Contained by:** module `galacticus_nodes`

**subroutine:** `basicnullbindingsetinteger0in`
**Description:** A null set function for rank 0 integers.
**Code lines:** 7
**Contained by:** module `galacticus_nodes`

**function:** `basicstandardaccretionrateget`
**Description:** Return the accretionRate property of the BasicStandard component implementation.
**Code lines:** 8
**Contained by:** module `galacticus_nodes`

**subroutine:** `basicstandardaccretionrateset`
**Description:** Set the accretionRate property of the BasicStandard component implementation.
**Code lines:** 9
**Contained by:** module `galacticus_nodes`
**Modules used:** `memory_management`

**function:** `basicstandardmasscount`
**Description:** Return a count of the number of scalar properties in the mass property of the basicStandard component implementation.
**Code lines:** 7
**Contained by:** module `galacticus_nodes`

**function:** `basicstandardmassget`
**Description:** Return the mass property of the BasicStandard component implementation.
**Code lines:** 8
19. Source Code Documentation

*Contained by:* module `galacticus_nodes`

**subroutine:** `basicstandardmassrate`

*Description:* Accumulate to the `mass` property rate of change of the `BasicStandard` component implementation.
*Code lines:* 10
*Contained by:* module `galacticus_nodes`

**subroutine:** `basicstandardmassscale`

*Description:* Set the `mass` property scale of the `BasicStandard` component implementation.
*Code lines:* 8
*Contained by:* module `galacticus_nodes`

**subroutine:** `basicstandardmassset`

*Description:* Set the `mass` property of the `BasicStandard` component implementation.
*Code lines:* 11
*Contained by:* module `galacticus_nodes`
*Modules used:* `memory_management`

**function:** `basicstandardtimecount`

*Description:* Return a count of the number of scalar properties in the `time` property of the `basicStandard` component implementation.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**function:** `basicstandardtimeget`

*Description:* Return the `time` property of the `BasicStandard` component implementation.
*Code lines:* 8
*Contained by:* module `galacticus_nodes`

**function:** `basicstandardtimelastisolated`

*Description:* Return the `timeLastIsolated` property of the `Basic` component class.
*Code lines:* 13
*Contained by:* module `galacticus_nodes`

**subroutine:** `basicstandardtimelastisolatedset`

*Description:* Set the `timeLastIsolated` property of the `BasicStandard` component implementation.
*Code lines:* 9
*Contained by:* module `galacticus_nodes`
*Modules used:* `memory_management`

**subroutine:** `basicstandardtimerate`

*Description:* Accumulate to the `time` property rate of change of the `BasicStandard` component implementation.
*Code lines:* 10
*Contained by:* module `galacticus_nodes`

**subroutine:** `basicstandardtimescale`

*Description:* Set the `time` property scale of the `BasicStandard` component implementation.
*Code lines:* 8
*Contained by:* module `galacticus_nodes`
19.1. Program units

**subroutine**: basicstandardtimeset  
*Description*: Set the *time* property of the *BasicStandard* component implementation.  
*Code lines*: 11  
*Contained by*: module *galacticus_nodes*  
*Modules used*: memory_management

**function**: basicstandardtrackingmassmaximumget  
*Description*: Return the *massMaximum* property of the *BasicStandardTracking* component implementation.  
*Code lines*: 8  
*Contained by*: module *galacticus_nodes*

**subroutine**: basicstandardtrackingmassmaximumset  
*Description*: Set the *massMaximum* property of the *BasicStandardTracking* component implementation.  
*Code lines*: 9  
*Contained by*: module *galacticus_nodes*  
*Modules used*: memory_management

**function**: basictime  
*Description*: Returns the default value for the *time* property for the *basic* component class.  
*Code lines*: 8  
*Contained by*: module *galacticus_nodes*

**function**: basictimeattributematch  
*Description*: Return a text list of component implementations in the *basic* class that have the desired attributes for the *time* property  
*Code lines*: 34  
*Contained by*: module *galacticus_nodes*  
*Modules used*: iso_varying_string

**function**: basictimeisgettable  
*Description*: Returns true if the *time* property is gettable for the *basic* component class.  
*Code lines*: 6  
*Contained by*: module *galacticus_nodes*

**function**: basictimelastisolated  
*Description*: Returns the default value for the *timeLastIsolated* property for the *basic* component class.  
*Code lines*: 8  
*Contained by*: module *galacticus_nodes*

**function**: basictimelastisolatedattributematch  
*Description*: Return a text list of component implementations in the *basic* class that have the desired attributes for the *timeLastIsolated* property  
*Code lines*: 38  
*Contained by*: module *galacticus_nodes*  
*Modules used*: iso_varying_string

**function**: basictimelastisolatedisgettable
19. Source Code Documentation

**Description:** Returns true if the `timeLastIsolated` property is gettable for the `basic` component class.

**Code lines:** 6

**Contained by:** module `galacticus_nodes`

**function: blackholeaccretionrate**

**Description:** Returns the default value for the `accretionRate` property for the `blackHole` component class.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**function: blackholeaccretionrateattributematch**

**Description:** Return a text list of component implementations in the `blackHole` class that have the desired attributes for the `accretionRate` property.

**Code lines:** 27

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function: blackholeaccretionrateisgettable**

**Description:** Returns true if the `accretionRate` property is gettable for the `blackHole` component class.

**Code lines:** 6

**Contained by:** module `galacticus_nodes`

**function: blackholecountlinked**

**Description:** Returns the number of `blackHole` components in `self`.

**Code lines:** 19

**Contained by:** module `galacticus_nodes`

**subroutine: blackholecreatebyinterrupt**

**Description:** Create the `blackHole` component of `self` via an interrupt.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**subroutine: blackholecreatelinked**

**Description:** Create the `blackHole` component of `self`.

**Code lines:** 29

**Contained by:** module `galacticus_nodes`

**Modules used:** `galacticus_display` `iso_varying_string` `string_handling`

**subroutine: blackholedestroylinked**

**Description:** Destroy the `blackHole` component of `self`.

**Code lines:** 13

**Contained by:** module `galacticus_nodes`

**function: blackholeget**

**Description:** Returns the `blackHole` component of `self`.

**Code lines:** 27

**Contained by:** module `galacticus_nodes`

**Modules used:** `galacticus_error`
function: blackholemass
Description: Returns the default value for the mass property for the blackHole component class.
Code lines: 16
Contained by: module galacticus_nodes

function: blackholemassattributematch
Description: Return a text list of component implementations in the blackHole class that have the desired attributes for the mass property
Code lines: 34
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: blackholemassisgettable
Description: Returns true if the mass property is gettable for the blackHole component class.
Code lines: 8
Contained by: module galacticus_nodes

function: blackholemassseed
Description: Returns the default value for the massSeed property for the blackHole component class.
Code lines: 8
Contained by: module galacticus_nodes

function: blackholemassseedattributematch
Description: Return a text list of component implementations in the blackHole class that have the desired attributes for the massSeed property
Code lines: 38
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: blackholemassseedisgettable
Description: Returns true if the massSeed property is gettable for the blackHole component class.
Code lines: 6
Contained by: module galacticus_nodes

function: blackholenullbindingdouble0inout
Description: A null get function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

function: blackholenullbindinginteger0in
Description: A null get function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: blackholenullbindingratedouble0inout
Description: A null rate function for rank 0 double precisions.
Code lines: 9
Contained by: module galacticus_nodes
**19. Source Code Documentation**

**subroutine**: blackholenullbindingrateinteger0in

*Description*: A null rate function for rank 0 integers.

*Code lines*: 9

*Contained by*: module *galacticus_nodes*

**subroutine**: blackholenullbindingsetdouble0inout

*Description*: A null set function for rank 0 double precisions.

*Code lines*: 7

*Contained by*: module *galacticus_nodes*

**subroutine**: blackholenullbindingsetinteger0in

*Description*: A null set function for rank 0 integers.

*Code lines*: 7

*Contained by*: module *galacticus_nodes*

**function**: blackholeradialposition

*Description*: Returns the default value for the *radialPosition* property for the *blackHole* component class.

*Code lines*: 8

*Contained by*: module *galacticus_nodes*

**function**: blackholeradialpositionattributematch

*Description*: Return a text list of component implementations in the *blackHole* class that have the desired attributes for the *radialPosition* property

*Code lines*: 25

*Contained by*: module *galacticus_nodes*

*Modules used*: *iso_varying_string*

**function**: blackholeradialpositionisgettable

*Description*: Returns true if the *radialPosition* property is gettable for the *blackHole* component class.

*Code lines*: 6

*Contained by*: module *galacticus_nodes*

**function**: blackholeradiativeefficiency

*Description*: Returns the default value for the *radiativeEfficiency* property for the *blackHole* component class.

*Code lines*: 8

*Contained by*: module *galacticus_nodes*

**function**: blackholeradiativeefficiencyattributematch

*Description*: Return a text list of component implementations in the *blackHole* class that have the desired attributes for the *radiativeEfficiency* property

*Code lines*: 27

*Contained by*: module *galacticus_nodes*

*Modules used*: *iso_varying_string*

**function**: blackholeradiativeefficiencyisgettable

*Description*: Returns true if the *radiativeEfficiency* property is gettable for the *blackHole* component class.
19.1. Program units

Code lines: 6
Contained by: module galacticus_nodes

function: blackholesimplemasscount
Description: Return a count of the number of scalar properties in the mass property of the blackHoleSimple component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: blackholesimplemassget
Description: Return the mass property of the BlackHoleSimple component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: blackholesimplemassrate
Description: Accumulate to the mass property rate of change of the BlackHoleSimple component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: blackholesimplemassscale
Description: Set the mass property scale of the BlackHoleSimple component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: blackholesimplemassset
Description: Set the mass property of the BlackHoleSimple component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: blackholespin
Description: Returns the default value for the spin property for the blackHole component class.
Code lines: 12
Contained by: module galacticus_nodes

function: blackholespinattributematch
Description: Return a text list of component implementations in the blackHole class that have the desired attributes for the spin property.
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: blackholespinisgettable
Description: Returns true if the spin property is gettable for the blackHole component class.
Code lines: 7
Contained by: module galacticus_nodes

function: blackholespinseed
19. Source Code Documentation

**Description:** Returns the default value for the `spinSeed` property for the `blackHole` component class.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**function:** `blackholespinseedattributematch`

**Description:** Return a text list of component implementations in the `blackHole` class that have the desired attributes for the `spinSeed` property.

**Code lines:** 27

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function:** `blackholespinseedisgettable`

**Description:** Returns true if the `spinSeed` property is gettable for the `blackHole` component class.

**Code lines:** 6

**Contained by:** module `galacticus_nodes`

**function:** `blackholestandardaccretionrateget`

**Description:** Get the value of the `accretionRate` property of the `BlackHoleStandard` component using a deferred function.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**subroutine:** `blackholestandardaccretionrategetfunction`

**Description:** Set the function to be used for get of the `accretionRate` property of the `BlackHoleStandard` component class.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**function:** `blackholestandardaccretionrategetisattached`

**Description:** Return true if the deferred function used to get the `accretionRate` property of the `BlackHoleStandard` component class has been attached.

**Code lines:** 5

**Contained by:** module `galacticus_nodes`

**function:** `blackholestandardmasscount`

**Description:** Return a count of the number of scalar properties in the `mass` property of the `blackHoleStandard` component implementation.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**function:** `blackholestandardmassget`

**Description:** Return the `mass` property of the `BlackHoleStandard` component implementation.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**subroutine:** `blackholestandardmassrate`

**Description:** Accumulate to the `mass` property rate of change of the `BlackHoleStandard` component implementation.

**Code lines:** 10

**Contained by:** module `galacticus_nodes`
19.1. Program units

**subroutine:** blackholestandardmassscale
*Description:* Set the mass property scale of the BlackHoleStandard component implementation.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine:** blackholestandardmassset
*Description:* Set the mass property of the BlackHoleStandard component implementation.
*Code lines:* 11
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**function:** blackholestandardradialpositioncount
*Description:* Return a count of the number of scalar properties in the radialPosition property of the BlackHoleStandard component implementation.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**function:** blackholestandardradialpositionget
*Description:* Return the radialPosition property of the BlackHoleStandard component implementation.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine:** blackholestandardradialpositionrate
*Description:* Accumulate to the radialPosition property rate of change of the BlackHoleStandard component implementation.
*Code lines:* 10
*Contained by:* module galacticus_nodes

**subroutine:** blackholestandardradialpositionscale
*Description:* Set the radialPosition property scale of the BlackHoleStandard component implementation.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine:** blackholestandardradialpositionset
*Description:* Set the radialPosition property of the BlackHoleStandard component implementation.
*Code lines:* 11
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**function:** blackholestandardradiativeefficiencyget
*Description:* Get the value of the radiativeEfficiency property of the BlackHoleStandard component using a deferred function.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine:** blackholestandardradiativeefficiencygetfunction
*Description:* Set the function to be used for get of the radiativeEfficiency property of the BlackHoleStandard component class.
function: blackholestandardradiativeefficiencygetisattached
Description: Return true if the deferred function used to get the radiativeEfficiency property of the BlackHoleStandard component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes

function: blackholestandardspincount
Description: Return a count of the number of scalar properties in the spin property of the blackHoleStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: blackholestandardspinrate
Description: Accumulate to the spin property rate of change of the BlackHoleStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: blackholestandardspinscale
Description: Set the spin property scale of the BlackHoleStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: blackholestandardspinset
Description: Set the spin property of the BlackHoleStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: blackholestandardtripleinteractiontimeget
Description: Return the tripleInteractionTime property of the BlackHoleStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: blackholestandardtripleinteractiontimeset
Description: Set the tripleInteractionTime property of the BlackHoleStandard component implementation.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

function: blackholetripleinteractiontime
Description: Returns the default value for the tripleInteractionTime property for the blackHole component class.
Code lines: 8
Contained by: module galacticus_nodes
function: blackholetripleinteractiontimeattributematch
Description: Return a text list of component implementations in the blackHole class that have the desired attributes for the tripleInteractionTime property
Code lines: 27
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: blackholetripleinteractiontimeisgettable
Description: Returns true if the tripleInteractionTime property is gettable for the blackHole component class.
Code lines: 6
Contained by: module galacticus_nodes

function: boolean_false
Description: Returns Boolean false always.
Code lines: 6
Contained by: module galacticus_nodes

function: boolean_true
Description: Returns Boolean true always.
Code lines: 6
Contained by: module galacticus_nodes

function: darkmatterprofilecountlinked
Description: Returns the number of darkMatterProfile components in self.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: darkmatterprofilecreatebyinterrupt
Description: Create the darkMatterProfile component of self via an interrupt.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: darkmatterprofilecreatelinked
Description: Create the darkMatterProfile component of self.
Code lines: 29
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: darkmatterprofiledestroylinked
Description: Destroy the darkMatterProfile component of self.
Code lines: 13
Contained by: module galacticus_nodes

function: darkmatterprofileget
Description: Returns the darkMatterProfile component of self.
Code lines: 27
function: darkmatterprofilenullbindingdouble0inout
Description: A null get function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

function: darkmatterprofilenullbindinginteger0in
Description: A null get function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: darkmatterprofilenullbindingratedouble0inout
Description: A null rate function for rank 0 double precisions.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: darkmatterprofilenullbindingrateinteger0in
Description: A null rate function for rank 0 integers.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: darkmatterprofilenullbindingsetdouble0inout
Description: A null set function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: darkmatterprofilenullbindingsetinteger0in
Description: A null set function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
Description: Returns the default value for the scaleGrowthRate property for the darkMatterProfile component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrate
19.1. Program units

Contained by:  module galacticus_nodes

function: darkmatterprofilescalegrowthrateattributematch
Description: Return a text list of component implementations in the darkMatterProfile class that have the desired attributes for the scaleGrowthRate property.
Code lines: 38
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: darkmatterprofilescalegrowthrateisgettable
Description: Returns true if the scaleGrowthRate property is gettable for the darkMatterProfile component class.
Code lines: 6
Contained by: module galacticus_nodes

function: darkmatterprofilescaleisgettable
Description: Returns true if the scale property is gettable for the darkMatterProfile component class.
Code lines: 6
Contained by: module galacticus_nodes

function: darkmatterprofilescalepresetscalecount
Description: Return a count of the number of scalar properties in the scale property of the darkMatterProfileScalePreset component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: darkmatterprofilescalepresetscaleget
Description: Return the scale property of the DarkMatterProfileScalePreset component implementation.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalepresetscalegrowthrateget
Description: Return the scaleGrowthRate property of the DarkMatterProfileScalePreset component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: darkmatterprofilescalepresetscalegrowthrateset
Description: Set the scaleGrowthRate property of the DarkMatterProfileScalePreset component implementation.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: darkmatterprofilescalepresetscalerate
Description: Accumulate to the scale property rate of change of the DarkMatterProfileScalePreset component implementation.
Code lines: 10
Contained by: module galacticus_nodes
subroutine: darkmatterprofilescalepresentscalescale
Description: Set the scale property scale of the DarkMatterProfileScalePreset component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: darkmatterprofilescalepresentscaleset
Description: Set the scale property of the DarkMatterProfileScalePreset component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: darkmatterprofilescalecount
Description: Return a count of the number of scalar properties in the scale property of the darkMatterProfileScale component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: darkmatterprofilescaleget
Description: Get the value of the scale property of the DarkMatterProfileScale component using a deferred function.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: darkmatterprofilescalegetfunction
Description: Set the function to be used for get of the scale property of the DarkMatterProfileScale component class.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegetisattached
Description: Return true if the deferred function used to get the scale property of the DarkMatterProfileScale component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes

function: darkmatterprofilescalegetvalue
Description: Return the scale property of the DarkMatterProfileScale component implementation.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescalegrowthrateget
Description: Return the scaleGrowthRate property of the DarkMatterProfileScale component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: darkmatterprofilescalegrowthrateset
Description: Set the scaleGrowthRate property of the DarkMatterProfileScale component implementation.
19.1. Program units

subroutine: darkmatterprofilescalescalerate
Description: Accumulate to the scale property rate of change of the Dark Matter Profile Scale component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: darkmatterprofilescalescalescale
Description: Set the scale property scale of the Dark Matter Profile Scale component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: darkmatterprofilescalescaleset
Description: Set the scale property of the Dark Matter Profile Scale component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: darkmatterprofilescaleshapeshapecount
Description: Return a count of the number of scalar properties in the shape property of the dark Matter Profile Scale Shape component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: darkmatterprofilescaleshapeshapeget
Description: Return the shape property of the Dark Matter Profile Scale Shape component implementation.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescaleshapeshapegrowthrateget
Description: Return the shape Growth Rate property of the Dark Matter Profile Scale Shape component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: darkmatterprofilescaleshapeshapegrowthrateset
Description: Set the shape Growth Rate property of the Dark Matter Profile Scale Shape component implementation.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: darkmatterprofilescaleshapeshaperate
Description: Accumulate to the shape property rate of change of the Dark Matter Profile Scale Shape component implementation.
Code lines: 10
19. Source Code Documentation

**subroutine:** darkmatterprofilescaleshapescaleshape

*Description:* Set the shape property scale of the DarkMatterProfileScaleShape component implementation.

*Code lines:* 8  
*Contained by:* module galacticus_nodes

**subroutine:** darkmatterprofilescaleshapeshape

*Description:* Set the shape property of the DarkMatterProfileScaleShape component implementation.

*Code lines:* 11  
*Contained by:* module galacticus_nodes

*Modules used:* memory_management

**function:** darkmatterprofileshape

*Description:* Returns the default value for the shape property for the darkMatterProfile component class.

*Code lines:* 8  
*Contained by:* module galacticus_nodes

**function:** darkmatterprofileshapeattributematch

*Description:* Return a text list of component implementations in the darkMatterProfile class that have the desired attributes for the shape property.

*Code lines:* 25  
*Contained by:* module galacticus_nodes

*Modules used:* iso_varying_string

**function:** darkmatterprofileshapegrowthrate

*Description:* Returns the default value for the shapeGrowthRate property for the darkMatterProfile component class.

*Code lines:* 8  
*Contained by:* module galacticus_nodes

**function:** darkmatterprofileshapegrowthrateattributematch

*Description:* Return a text list of component implementations in the darkMatterProfile class that have the desired attributes for the shapeGrowthRate property.

*Code lines:* 27  
*Contained by:* module galacticus_nodes

*Modules used:* iso_varying_string

**function:** darkmatterprofileshapegrowthrateisgettable

*Description:* Returns true if the shapeGrowthRate property is gettable for the darkMatterProfile component class.

*Code lines:* 6  
*Contained by:* module galacticus_nodes

**function:** darkmatterprofileshapeisgettable

*Description:* Returns true if the shape property is gettable for the darkMatterProfile component class.

*Code lines:* 6  
*Contained by:* module galacticus_nodes
subroutine: deserializefromarrayrates
  Description: Deserialize rates from array.
  Code lines: 307
  Contained by: module galacticus_nodes
  Modules used: memory_management

subroutine: deserializefromarrayscales
  Description: Deserialize scales from array.
  Code lines: 307
  Contained by: module galacticus_nodes
  Modules used: memory_management

subroutine: deserializefromarrayvalues
  Description: Deserialize values from array.
  Code lines: 307
  Contained by: module galacticus_nodes
  Modules used: memory_management

function: diskabundancesgas
  Description: Returns the default value for the abundancesGas property for the disk component class.
  Code lines: 8
  Contained by: module galacticus_nodes

function: diskabundancesgasattributematch
  Description: Return a text list of component implementations in the disk class that have the desired attributes for the abundancesGas property
  Code lines: 25
  Contained by: module galacticus_nodes
  Modules used: iso_varying_string

function: diskabundancesgasisgettable
  Description: Returns true if the abundancesGas property is gettable for the disk component class.
  Code lines: 6
  Contained by: module galacticus_nodes

subroutine: diskabundancesgasrate
  Description: Accept a rate set for the abundancesGas property of the disk component class. Trigger an interrupt to create the component.
  Code lines: 15
  Contained by: module galacticus_nodes
  Modules used: galacticus_error

function: diskabundancesstellar
  Description: Returns the default value for the abundancesStellar property for the disk component class.
  Code lines: 8
  Contained by: module galacticus_nodes

function: diskabundancesstellarattributematch
19. Source Code Documentation

Description: Return a text list of component implementations in the disk class that have the desired attributes for the abundancesStellar property
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: diskabundancesstellarisgettable
Description: Returns true if the abundancesStellar property is gettable for the disk component class.
Code lines: 6
Contained by: module galacticus_nodes

function: diskangularmomentum
Description: Returns the default value for the angularMomentum property for the disk component class.
Code lines: 8
Contained by: module galacticus_nodes

function: diskangularmomentumattributematch
Description: Return a text list of component implementations in the disk class that have the desired attributes for the angularMomentum property
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: diskangularmomentumisgettable
Description: Returns true if the angularMomentum property is gettable for the disk component class.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: diskangularmomentumrate
Description: Accept a rate set for the angularMomentum property of the disk component class. Trigger an interrupt to create the component.
Code lines: 15
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: diskcountlinked
Description: Returns the number of disk components in self.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: diskcreatelyinterrupt
Description: Create the disk component of self via an interrupt.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: diskcreatelinked
Description: Create the disk component of self.
Code lines: 29
Contained by: module galacticus_nodes
19.1. Program units

subroutine: diskdestroylinked
Description: Destroy the disk component of self.
Code lines: 13
Contained by: module galacticus_nodes

function: diskexponentialabundancesgascount
Description: Return a count of the number of scalar properties in the abundancesGas property of the diskExponential component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: diskexponentialabundancesgasget
Description: Return the abundancesGas property of the DiskExponential component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: diskexponentialabundancesgasrate
Description: Accumulate to the abundancesGas property rate of change of the DiskExponential component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: diskexponentialabundancesgasrategeneric
Description: Set the rate of the abundancesGas property of the DiskExponential component via a generic nodeComponent.
Code lines: 24
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: diskexponentialabundancesgasscale
Description: Set the abundancesGas property scale of the DiskExponential component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: diskexponentialabundancesgasset
Description: Set the abundancesGas property of the DiskExponential component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: diskexponentialabundancesstellarcount
Description: Return a count of the number of scalar properties in the abundancesStellar property of the diskExponential component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: diskexponentialabundancesstellarget

Modules used: galacticus_display iso_varying_string string_handling

1401
Description: Return the `abundancesStellar` property of the `DiskExponential` component implementation.

Code lines: 8

Contained by: module `galacticus_nodes`

**subroutine: diskexponentialabundancesstellarrate**

Description: Accumulate to the `abundancesStellar` property rate of change of the `DiskExponential` component implementation.

Code lines: 10

Contained by: module `galacticus_nodes`

**subroutine: diskexponentialabundancesstellarscale**

Description: Set the `abundancesStellar` property scale of the `DiskExponential` component implementation.

Code lines: 8

Contained by: module `galacticus_nodes`

**subroutine: diskexponentialabundancesstellarset**

Description: Set the `abundancesStellar` property of the `DiskExponential` component implementation.

Code lines: 11

Contained by: module `galacticus_nodes`

Modules used: `memory_management`

**function: diskexponentialangularmomentumcount**

Description: Return a count of the number of scalar properties in the `angularMomentum` property of the `diskExponential` component implementation.

Code lines: 7

Contained by: module `galacticus_nodes`

**function: diskexponentialangularmomentumget**

Description: Return the `angularMomentum` property of the `DiskExponential` component implementation.

Code lines: 8

Contained by: module `galacticus_nodes`

**subroutine: diskexponentialangularmomentumrate**

Description: Accumulate to the `angularMomentum` property rate of change of the `DiskExponential` component implementation.

Code lines: 10

Contained by: module `galacticus_nodes`

**subroutine: diskexponentialangularmomentumrategeneric**

Description: Set the rate of the `angularMomentum` property of the `DiskExponential` component via a generic `nodeComponent`.

Code lines: 24

Contained by: module `galacticus_nodes`

Modules used: `galacticus_error`

**subroutine: diskexponentialangularmomentumscale**

Description: Set the `angularMomentum` property scale of the `DiskExponential` component implementation.

Code lines: 8
19.1. Program units

**subroutine:** diskexponentialangularmomentumset
*Description:* Set the angularMomentum property of the DiskExponential component implementation.
*Code lines:* 11
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**function:** diskexponentialisinitializedget
*Description:* Return the isInitialized property of the DiskExponential component implementation.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine:** diskexponentialisinitializedset
*Description:* Set the isInitialized property of the DiskExponential component implementation.
*Code lines:* 9
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**function:** diskexponentialluminositiesstellarcount
*Description:* Return a count of the number of scalar properties in the luminositiesStellar property of the diskExponential component implementation.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**function:** diskexponentialluminositiesstellarget
*Description:* Return the luminositiesStellar property of the DiskExponential component implementation.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine:** diskexponentialluminositiesstellarrate
*Description:* Accumulate to the luminositiesStellar property rate of change of the DiskExponential component implementation.
*Code lines:* 10
*Contained by:* module galacticus_nodes

**subroutine:** diskexponentialluminositiesstellarscale
*Description:* Set the luminositiesStellar property scale of the DiskExponential component implementation.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine:** diskexponentialluminositiesstellarset
*Description:* Set the luminositiesStellar property of the DiskExponential component implementation.
*Code lines:* 11
*Contained by:* module galacticus_nodes
*Modules used:* memory_management
function: diskexponentialmassgascount
Description: Return a count of the number of scalar properties in the massGas property of the diskExponential component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: diskexponentialmassgasget
Description: Return the massGas property of the DiskExponential component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: diskexponentialmassgasrate
Description: Accumulate to the massGas property rate of change of the DiskExponential component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: diskexponentialmassgasrategeneric
Description: Set the rate of the massGas property of the DiskExponential component via a generic nodeComponent.
Code lines: 24
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: diskexponentialmassgasscale
Description: Set the massGas property scale of the DiskExponential component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: diskexponentialmassgasset
Description: Set the massGas property of the DiskExponential component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: diskexponentialmassstellarcount
Description: Return a count of the number of scalar properties in the massStellar property of the diskExponential component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: diskexponentialmassstellarget
Description: Return the massStellar property of the DiskExponential component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: diskexponentialmassstellarrate
Description: Accumulate to the massStellar property rate of change of the DiskExponential component implementation.
19.1. Program units

Subroutine: diskexponentialmassstellarscale
   Description: Set the massStellar property scale of the DiskExponential component implementation.
   Code lines: 8
   Contained by: module galacticus_nodes

Subroutine: diskexponentialmassstellarset
   Description: Set the massStellar property of the DiskExponential component implementation.
   Code lines: 11
   Contained by: module galacticus_nodes
   Modules used: memory_management

Function: diskexponentialradiusget
   Description: Return the radius property of the DiskExponential component implementation.
   Code lines: 8
   Contained by: module galacticus_nodes

Subroutine: diskexponentialradiusset
   Description: Set the radius property of the DiskExponential component implementation.
   Code lines: 9
   Contained by: module galacticus_nodes
   Modules used: memory_management

Function: diskexponentialstarformationhistorycount
   Description: Return a count of the number of scalar properties in the starFormationHistory property of the diskExponential component implementation.
   Code lines: 7
   Contained by: module galacticus_nodes

Function: diskexponentialstarformationhistoryget
   Description: Return the starFormationHistory property of the DiskExponential component implementation.
   Code lines: 8
   Contained by: module galacticus_nodes

Subroutine: diskexponentialstarformationhistoryrate
   Description: Accumulate to the starFormationHistory property rate of change of the DiskExponential component implementation.
   Code lines: 10
   Contained by: module galacticus_nodes

Subroutine: diskexponentialstarformationhistoryscale
   Description: Set the starFormationHistory property scale of the DiskExponential component implementation.
   Code lines: 8
   Contained by: module galacticus_nodes

Subroutine: diskexponentialstarformationhistoryset
19. Source Code Documentation

**Description:** Set the `starFormationHistory` property of the `DiskExponential` component implementation.

**Code lines:** 11

**Contained by:** module `galacticus_nodes`

**Modules used:** `memory_management`

**function:** `diskexponentialstarformationrateget`

**Description:** Get the value of the `starFormationRate` property of the `DiskExponential` component using a deferred function.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**subroutine:** `diskexponentialstarformationrategetfunction`

**Description:** Set the function to be used for get of the `starFormationRate` property of the `DiskExponential` component class.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**function:** `diskexponentialstarformationrategetisattached`

**Description:** Return true if the deferred function used to get the `starFormationRate` property of the `DiskExponential` component class has been attached.

**Code lines:** 5

**Contained by:** module `galacticus_nodes`

**function:** `diskexponentialstellarpropertieshistorycount`

**Description:** Return a count of the number of scalar properties in the `stellarPropertiesHistory` property of the `diskExponential` component implementation.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**function:** `diskexponentialstellarpropertieshistoryget`

**Description:** Return the `stellarPropertiesHistory` property of the `DiskExponential` component implementation.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**subroutine:** `diskexponentialstellarpropertieshistoryrate`

**Description:** Accumulate to the `stellarPropertiesHistory` property rate of change of the `DiskExponential` component implementation.

**Code lines:** 10

**Contained by:** module `galacticus_nodes`

**subroutine:** `diskexponentialstellarpropertieshistoryscale`

**Description:** Set the `stellarPropertiesHistory` property scale of the `DiskExponential` component implementation.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**subroutine:** `diskexponentialstellarpropertieshistoryset`

**Description:** Set the `stellarPropertiesHistory` property of the `DiskExponential` component implementation.
19.1. Program units

Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

**function: diskexponentialvelocityget**  
*Description:* Return the velocity property of the DiskExponential component implementation.  
*Code lines:* 8  
*Contained by:* module galacticus_nodes

**subroutine: diskexponentialvelocityset**  
*Description:* Set the velocity property of the DiskExponential component implementation.  
*Code lines:* 9  
*Contained by:* module galacticus_nodes

**function: diskget**  
*Description:* Returns the disk component of self.  
*Code lines:* 27  
*Contained by:* module galacticus_nodes

**function: diskhalfmassradius**  
*Description:* Returns the default value for the halfMassRadius property for the disk component class.  
*Code lines:* 8  
*Contained by:* module galacticus_nodes

**function: diskhalfmassradiusattributematch**  
*Description:* Return a text list of component implementations in the disk class that have the desired attributes for the halfMassRadius property  
*Code lines:* 27  
*Contained by:* module galacticus_nodes

**function: diskhalfmassradiusisgettable**  
*Description:* Returns true if the halfMassRadius property is gettable for the disk component class.  
*Code lines:* 6  
*Contained by:* module galacticus_nodes

**function: diskisinitialized**  
*Description:* Returns the default value for the isInitialized property for the disk component class.  
*Code lines:* 8  
*Contained by:* module galacticus_nodes

**function: diskisinitializedattributematch**  
*Description:* Return a text list of component implementations in the disk class that have the desired attributes for the isInitialized property  
*Code lines:* 27  
*Contained by:* module galacticus_nodes

**Modules used:**
- memory_management
function: diskisinitializedisgettable
Description: Returns true if the isInitialized property is gettable for the disk component class.
Code lines: 6
Contained by: module galacticus_nodes

function: diskluminositiesstellar
Description: Returns the default value for the luminositiesStellar property for the disk component class.
Code lines: 8
Contained by: module galacticus_nodes

function: diskluminositiesstellarattributematch
Description: Return a text list of component implementations in the disk class that have the desired attributes for the luminositiesStellar property
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: diskluminositiesstellarisgettable
Description: Returns true if the luminositiesStellar property is gettable for the disk component class.
Code lines: 6
Contained by: module galacticus_nodes

function: diskmassgas
Description: Returns the default value for the massGas property for the disk component class.
Code lines: 8
Contained by: module galacticus_nodes

function: diskmassgasattributematch
Description: Return a text list of component implementations in the disk class that have the desired attributes for the massGas property
Code lines: 34
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: diskmassgasisgettable
Description: Returns true if the massGas property is gettable for the disk component class.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: diskmassgasrate
Description: Accept a rate set for the massGas property of the disk component class. Trigger an interrupt to create the component.
Code lines: 15
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: diskmassstellar
Description: Returns the default value for the massStellar property for the disk component class.
19.1. Program units

function: diskmassstellarattributematch
Description: Return a text list of component implementations in the disk class that have the desired attributes for the massStellar property.
Code lines: 34
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: diskmassstellarisgettable
Description: Returns true if the massStellar property is gettable for the disk component class.
Code lines: 6
Contained by: module galacticus_nodes

function: disknullbindingabundances0inout
Description: A null get function for rank 0 typeabundances.
Code lines: 7
Contained by: module galacticus_nodes

function: disknullbindingdouble0inout
Description: A null get function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

function: disknullbindinghistory0inout
Description: A null get function for rank 0 typehistories.
Code lines: 7
Contained by: module galacticus_nodes

function: disknullbindinginteger0in
Description: A null get function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

function: disknullbindinglogical0inout
Description: A null get function for rank 0 logicals.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: disknullbindingrateabundances0inout
Description: A null rate function for rank 0 typeabundances.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: disknullbindingratedouble0inout
Description: A null rate function for rank 0 double precisions.
Code lines: 9
Contained by: module galacticus_nodes
subroutine: disknullbindingratehistory0inout
   Description: A null rate function for rank 0 typehistorys.
   Code lines: 9
   Contained by: module galacticus_nodes

subroutine: disknullbindingrateinteger0in
   Description: A null rate function for rank 0 integers.
   Code lines: 9
   Contained by: module galacticus_nodes

subroutine: disknullbindingratelogical0inout
   Description: A null rate function for rank 0 logicals.
   Code lines: 9
   Contained by: module galacticus_nodes

subroutine: disknullbindingratestellarluminosities0inout
   Description: A null rate function for rank 0 typestellarluminositiess.
   Code lines: 9
   Contained by: module galacticus_nodes

subroutine: disknullbindingsetabundances0inout
   Description: A null set function for rank 0 typeabundances.
   Code lines: 7
   Contained by: module galacticus_nodes

subroutine: disknullbindingsetdouble0inout
   Description: A null set function for rank 0 double precisions.
   Code lines: 7
   Contained by: module galacticus_nodes

subroutine: disknullbindingsethistory0inout
   Description: A null set function for rank 0 typehistorys.
   Code lines: 7
   Contained by: module galacticus_nodes

subroutine: disknullbindingsetinteger0in
   Description: A null set function for rank 0 integers.
   Code lines: 7
   Contained by: module galacticus_nodes

subroutine: disknullbindingsetlogical0inout
   Description: A null set function for rank 0 logicals.
   Code lines: 7
   Contained by: module galacticus_nodes

subroutine: disknullbindingsetstellarluminosities0inout
   Description: A null set function for rank 0 typestellarluminositiess.
   Code lines: 7
19.1. Program units

Contained by: module galacticus_nodes

function: disknullbindingstellarluminosities0inout
Description: A null get function for rank 0 typestellarluminositiess.
Code lines: 7
Contained by: module galacticus_nodes

function: diskradius
Description: Returns the default value for the radius property for the disk component class.
Code lines: 8
Contained by: module galacticus_nodes

function: diskradiusattributematch
Description: Return a text list of component implementations in the disk class that have the desired attributes for the radius property.
Code lines: 27
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: diskradiusisgettable
Description: Returns true if the radius property is gettable for the disk component class.
Code lines: 6
Contained by: module galacticus_nodes

function: diskstarformationhistory
Description: Returns the default value for the starFormationHistory property for the disk component class.
Code lines: 8
Contained by: module galacticus_nodes

function: diskstarformationhistoryattributematch
Description: Return a text list of component implementations in the disk class that have the desired attributes for the starFormationHistory property.
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: diskstarformationhistoryisgettable
Description: Returns true if the starFormationHistory property is gettable for the disk component class.
Code lines: 6
Contained by: module galacticus_nodes

function: diskstarformationrate
Description: Returns the default value for the starFormationRate property for the disk component class.
Code lines: 8
Contained by: module galacticus_nodes

function: diskstarformationrateattributematch
Description: Return a text list of component implementations in the disk class that have the desired attributes for the starFormationRate property

Code lines: 27

Contained by: module galacticus_nodes

Modules used: iso_varying_string

function: diskstarformationrateisgettable

Description: Returns true if the starFormationRate property is gettable for the disk component class.

Code lines: 6

Contained by: module galacticus_nodes

function: diskstellarpropertieshistory

Description: Returns the default value for the stellarPropertiesHistory property for the disk component class.

Code lines: 8

Contained by: module galacticus_nodes

function: diskstellarpropertieshistoryattributematch

Description: Return a text list of component implementations in the disk class that have the desired attributes for the stellarPropertiesHistory property

Code lines: 25

Contained by: module galacticus_nodes

Modules used: iso_varying_string

function: diskstellarpropertieshistoryisgettable

Description: Returns true if the stellarPropertiesHistory property is gettable for the disk component class.

Code lines: 6

Contained by: module galacticus_nodes

function: diskvelocity

Description: Returns the default value for the velocity property for the disk component class.

Code lines: 8

Contained by: module galacticus_nodes

function: diskvelocityattributematch

Description: Return a text list of component implementations in the disk class that have the desired attributes for the velocity property

Code lines: 27

Contained by: module galacticus_nodes

Modules used: iso_varying_string

function: diskvelocityisgettable

Description: Returns true if the velocity property is gettable for the disk component class.

Code lines: 6

Contained by: module galacticus_nodes

function: diskverysimplemassgascount

Description: Return a count of the number of scalar properties in the massGas property of the diskVerySimple component implementation.
19.1. Program units

**Code lines:** 7  
**Contained by:** module `galacticus_nodes`

**function:** `diskverysimplemassgasset`  
**Description:** Return the `massGas` property of the `DiskVerySimple` component implementation.  
**Code lines:** 8  
**Contained by:** module `galacticus_nodes`

**subroutine:** `diskverysimplemassgasrate`  
**Description:** Accumulate to the `massGas` property rate of change of the `DiskVerySimple` component implementation.  
**Code lines:** 10  
**Contained by:** module `galacticus_nodes`

**subroutine:** `diskverysimplemassgasrategeneric`  
**Description:** Set the rate of the `massGas` property of the `DiskVerySimple` component via a generic `nodeComponent`.  
**Code lines:** 24  
**Contained by:** module `galacticus_nodes`  
**Modules used:** `galacticus_error`

**subroutine:** `diskverysimplemassgasscale`  
**Description:** Set the `massGas` property scale of the `DiskVerySimple` component implementation.  
**Code lines:** 8  
**Contained by:** module `galacticus_nodes`

**subroutine:** `diskverysimplemassgasset`  
**Description:** Set the `massGas` property of the `DiskVerySimple` component implementation.  
**Code lines:** 11  
**Contained by:** module `galacticus_nodes`  
**Modules used:** `memory_management`

**function:** `diskverysimplemassstellarcount`  
**Description:** Return a count of the number of scalar properties in the `massStellar` property of the `diskVerySimple` component implementation.  
**Code lines:** 7  
**Contained by:** module `galacticus_nodes`

**function:** `diskverysimplemassstellarget`  
**Description:** Return the `massStellar` property of the `DiskVerySimple` component implementation.  
**Code lines:** 8  
**Contained by:** module `galacticus_nodes`

**subroutine:** `diskverysimplemassstellarrate`  
**Description:** Accumulate to the `massStellar` property rate of change of the `DiskVerySimple` component implementation.  
**Code lines:** 10  
**Contained by:** module `galacticus_nodes`

**subroutine:** `diskverysimplemassstellarscale`
19. Source Code Documentation

**Description:** Set the `massStellar` property scale of the `DiskVerySimple` component implementation.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**subroutine: diskverysimplemassstellarset**

**Description:** Set the `massStellar` property of the `DiskVerySimple` component implementation.

**Code lines:** 11

**Contained by:** module `galacticus_nodes`

**Modules used:** `memory_management`

**function: dynamicsstatisticsadiabaticratio**

**Description:** Returns the default value for the `adiabaticRatio` property for the `dynamicsStatistics` component class.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**function: dynamicsstatisticsadiabaticratioattributematch**

**Description:** Return a text list of component implementations in the `dynamicsStatistics` class that have the desired attributes for the `adiabaticRatio` property.

**Code lines:** 27

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function: dynamicsstatisticsadiabaticratioisgettable**

**Description:** Returns true if the `adiabaticRatio` property is gettable for the `dynamicsStatistics` component class.

**Code lines:** 6

**Contained by:** module `galacticus_nodes`

**function: dynamicsstatisticsbarinstabilitytimescale**

**Description:** Returns the default value for the `barInstabilityTimescale` property for the `dynamicsStatistics` component class.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**function: dynamicsstatisticsbarinstabilitytimescaleattributematch**

**Description:** Return a text list of component implementations in the `dynamicsStatistics` class that have the desired attributes for the `barInstabilityTimescale` property.

**Code lines:** 27

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function: dynamicsstatisticsbarinstabilitytimescaleisgettable**

**Description:** Returns true if the `barInstabilityTimescale` property is gettable for the `dynamicsStatistics` component class.

**Code lines:** 6

**Contained by:** module `galacticus_nodes`

**function: dynamicsstatisticsbarsadiabaticratioget**
19.1. Program units

Description: Return the adiabaticRatio property of the DynamicsStatisticsBars component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: dynamicsstatisticsbarsadiabaticratioset
Description: Set the adiabaticRatio property of the DynamicsStatisticsBars component implementation.
Code lines: 17
Contained by: module galacticus_nodes
Modules used: memory_management

function: dynamicsstatisticsbarsbarinstabilitytimescaleget
Description: Return the barInstabilityTimescale property of the DynamicsStatisticsBars component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: dynamicsstatisticsbarsbarinstabilitytimescaleset
Description: Set the barInstabilityTimescale property of the DynamicsStatisticsBars component implementation.
Code lines: 17
Contained by: module galacticus_nodes
Modules used: memory_management

function: dynamicsstatisticsbarstimeget
Description: Return the time property of the DynamicsStatisticsBars component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: dynamicsstatisticsbarstimeset
Description: Set the time property of the DynamicsStatisticsBars component implementation.
Code lines: 17
Contained by: module galacticus_nodes
Modules used: memory_management

function: dynamicsstatisticscountlinked
Description: Returns the number of dynamicsStatistics components in self.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: dynamicsstatisticscreatebyinterrupt
Description: Create the dynamicsStatistics component of self via an interrupt.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: dynamicsstatisticscreatelinked
Description: Create the dynamicsStatistics component of self.
Code lines: 29
19. Source Code Documentation

**Contained by:** module galacticus_nodes

**Modules used:**
- galacticus_display
- iso_varying_string
- string_handling

**subroutine:** dynamicsstatisticsdestroylinked

*Description:* Destroy the dynamicsStatistics component of self.
*Code lines:* 13
*Contained by:* module galacticus_nodes

**function:** dynamicsstatisticsget

*Description:* Returns the dynamicsStatistics component of self.
*Code lines:* 27
*Contained by:* module galacticus_nodes

**Modules used:** galacticus_error

**function:** dynamicsstatisticsnullbindingdouble1inout

*Description:* A null get function for rank 1 double precisions.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**function:** dynamicsstatisticsnullbindinginteger0in

*Description:* A null get function for rank 0 integers.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** dynamicsstatisticsnullbindingratedouble1inout

*Description:* A null rate function for rank 1 double precisions.
*Code lines:* 9
*Contained by:* module galacticus_nodes

**subroutine:** dynamicsstatisticsnullbindingrateinteger0in

*Description:* A null rate function for rank 0 integers.
*Code lines:* 9
*Contained by:* module galacticus_nodes

**subroutine:** dynamicsstatisticsnullbindingsetdouble1inout

*Description:* A null set function for rank 1 double precisions.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** dynamicsstatisticsnullbindingsetinteger0in

*Description:* A null set function for rank 0 integers.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**function:** dynamicsstatisticstime

*Description:* Returns the default value for the time property for the dynamicsStatistics component class.
*Code lines:* 8
*Contained by:* module galacticus_nodes
19.1. Program units

function: dynamicsstatisticstimeattributematch
  Description: Return a text list of component implementations in the dynamicsStatistics class that have the desired attributes for the time property
  Code lines: 27
  Contained by: module galacticus_nodes
  Modules used: iso_varying_string

function: dynamicsstatisticstimeisgettable
  Description: Returns true if the time property is gettable for the dynamicsStatistics component class.
  Code lines: 6
  Contained by: module galacticus_nodes

function: formationtimecole2000formationtime
  Description: Return the formationTime property of the formationTime component class.
  Code lines: 19
  Contained by: module galacticus_nodes

function: formationtimecountlinked
  Description: Returns the number of formationTime components in self.
  Code lines: 19
  Contained by: module galacticus_nodes

subroutine: formationtimecreatebyinterrupt
  Description: Create the formationTime component of self via an interrupt.
  Code lines: 8
  Contained by: module galacticus_nodes

subroutine: formationtimecreatelinked
  Description: Create the formationTime component of self.
  Code lines: 29
  Contained by: module galacticus_nodes
  Modules used: galacticus_display iso_varying_string string_handling

subroutine: formationtimedestroylinked
  Description: Destroy the formationTime component of self.
  Code lines: 13
  Contained by: module galacticus_nodes

function: formationtimeformationtime
  Description: Returns the default value for the formationTime property for the formationTime component class.
  Code lines: 8
  Contained by: module galacticus_nodes

function: formationtimeformationtimeattributematch
  Description: Return a text list of component implementations in the formationTime class that have the desired attributes for the formationTime property
  Code lines: 27
  Contained by: module galacticus_nodes
19. Source Code Documentation

Modules used: iso_varying_string

function: formationtimeformationtimeisgettable
Description: Returns true if the formationTime property is gettable for the formationTime component class.
Code lines: 6
Contained by: module galacticus_nodes

function: formationtimeget
Description: Returns the formationTime component of self.
Code lines: 27
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: formationtimenullbindingdouble0inout
Description: A null get function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

function: formationtimenullbindinginteger0in
Description: A null get function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: formationtimenullbindingratedouble0inout
Description: A null rate function for rank 0 double precisions.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: formationtimenullbindingrateinteger0in
Description: A null rate function for rank 0 integers.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: formationtimenullbindingsetdouble0inout
Description: A null set function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: formationtimenullbindingsetinteger0in
Description: A null set function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: galacticus_nodes_finalize
Description: Finalize the GALACTICUS object system.
Code lines: 24
Contained by: module galacticus_nodes
19.1. Program units

subroutine: galacticus_nodes_initialize
  Description: Initialize the GALACTICUS object system.
  Code lines: 647
  Contained by: module galacticus_nodes
  Modules used: input_parameters iso_varying_string memory_management

subroutine: galacticus_nodes_unique_id_set
  Description: Resets the global unique ID number.
  Code lines: 7
  Contained by: module galacticus_nodes

function: genericnullbindingabundances0inout
  Description: A null get function for rank 0 type abundances.
  Code lines: 7
  Contained by: module galacticus_nodes

function: genericnullbindingdouble0inout
  Description: A null get function for rank 0 double precisions.
  Code lines: 7
  Contained by: module galacticus_nodes

subroutine: genericnullbindingrateabundances0inout
  Description: A null rate function for rank 0 type abundances.
  Code lines: 9
  Contained by: module galacticus_nodes

subroutine: genericnullbindingratedouble0inout
  Description: A null rate function for rank 0 double precisions.
  Code lines: 9
  Contained by: module galacticus_nodes

subroutine: genericnullbindingsetabundances0inout
  Description: A null set function for rank 0 type abundances.
  Code lines: 7
  Contained by: module galacticus_nodes

subroutine: genericnullbindingsetdouble0inout
  Description: A null set function for rank 0 double precisions.
  Code lines: 7
  Contained by: module galacticus_nodes

function: hosthistorycountlinked
  Description: Returns the number of hostHistory components in self.
  Code lines: 19
  Contained by: module galacticus_nodes
subroutine: hosthistorycreatebyinterrupt
Description: Create the hostHistory component of self via an interrupt.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hosthistorycreatelinked
Description: Create the hostHistory component of self.
Code lines: 29
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: hosthistorydestroylinked
Description: Destroy the hostHistory component of self.
Code lines: 13
Contained by: module galacticus_nodes

function: hosthistoryget
Description: Returns the hostHistory component of self.
Code lines: 27
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: hosthistoryhostmassmaximum
Description: Returns the default value for the hostMassMaximum property for the hostHistory component class.
Code lines: 12
Contained by: module galacticus_nodes

function: hosthistoryhostmassmaximumattributematch
Description: Return a text list of component implementations in the hostHistory class that have the desired attributes for the hostMassMaximum property
Code lines: 27
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: hosthistoryhostmassmaximumisgettable
Description: Returns true if the hostMassMaximum property is gettable for the hostHistory component class.
Code lines: 7
Contained by: module galacticus_nodes

function: hosthistorynullbindingdouble0inout
Description: A null get function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

function: hosthistorynullbindinginteger0in
19.1. Program units

Description: A null get function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: hosthistorynullbindingratedouble0inout
Description: A null rate function for rank 0 double precisions.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: hosthistorynullbindingrateinteger0in
Description: A null rate function for rank 0 integers.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: hosthistorynullbindingsetdouble0inout
Description: A null set function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: hosthistorynullbindingsetinteger0in
Description: A null set function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

function: hosthistorystandardhostmassmaximumget
Description: Return the hostMassMaximum property of the HostHistoryStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hosthistorystandardhostmassmaximumset
Description: Set the hostMassMaximum property of the HostHistoryStandard component implementation.
Code lines: 9
Contained by: module galacticus_nodes

Modules used: memory_management

function: hothaloabundances
Description: Returns the default value for the abundances property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothaloabundancesattributematch
Description: Return a text list of component implementations in the hotHalo class that have the desired attributes for the abundances property
Code lines: 25
Contained by: module galacticus_nodes

Modules used: iso_varying_string

function: hothaloabundancescold
19. Source Code Documentation

Description: Returns the default value for the \texttt{abundancesCold} property for the \texttt{hotHalo} component class.
Code lines: 8
Contained by: module \texttt{galacticus\_nodes}

\textbf{function: hothaloabundancescoldattributematch}

Description: Return a text list of component implementations in the \texttt{hotHalo} class that have the desired attributes for the \texttt{abundancesCold} property.
Code lines: 25
Contained by: module \texttt{galacticus\_nodes}
Modules used: \texttt{iso\_varying\_string}

\textbf{function: hothaloabundancescoldisgettable}

Description: Returns true if the \texttt{abundancesCold} property is gettable for the \texttt{hotHalo} component class.
Code lines: 6
Contained by: module \texttt{galacticus\_nodes}

\textbf{subroutine: hothaloabundancescoldrate}

Description: Accept a rate set for the \texttt{abundancesCold} property of the \texttt{hotHalo} component class. Trigger an interrupt to create the component.
Code lines: 15
Contained by: module \texttt{galacticus\_nodes}
Modules used: \texttt{galacticus\_error}

\textbf{function: hothaloabundancesisgettable}

Description: Returns true if the \texttt{abundances} property is gettable for the \texttt{hotHalo} component class.
Code lines: 6
Contained by: module \texttt{galacticus\_nodes}

\textbf{subroutine: hothaloabundancesrate}

Description: Accept a rate set for the \texttt{abundances} property of the \texttt{hotHalo} component class. Trigger an interrupt to create the component.
Code lines: 15
Contained by: module \texttt{galacticus\_nodes}
Modules used: \texttt{galacticus\_error}

\textbf{function: hothaloangularmomentum}

Description: Returns the default value for the \texttt{angularMomentum} property for the \texttt{hotHalo} component class.
Code lines: 8
Contained by: module \texttt{galacticus\_nodes}

\textbf{function: hothaloangularmomentumattributematch}

Description: Return a text list of component implementations in the \texttt{hotHalo} class that have the desired attributes for the \texttt{angularMomentum} property.
Code lines: 25
Contained by: module \texttt{galacticus\_nodes}
Modules used: \texttt{iso\_varying\_string}

\textbf{function: hothaloangularmomentumcold}
Description: Returns the default value for the \texttt{angular MomentumCold} property for the \texttt{hotHalo} component class.

Code lines: 8

Contained by: module \texttt{galacticus\_nodes}

\textbf{function: hothalo\_angular\_momentum\_cold\_attributematch}

Description: Return a text list of component implementations in the \texttt{hotHalo} class that have the desired attributes for the \texttt{angular MomentumCold} property.

Code lines: 25

Contained by: module \texttt{galacticus\_nodes}

Modules used: \texttt{iso\_varying\_string}

\textbf{function: hothalo\_angular\_momentum\_cold\_isgettable}

Description: Returns true if the \texttt{angular MomentumCold} property is gettable for the \texttt{hotHalo} component class.

Code lines: 6

Contained by: module \texttt{galacticus\_nodes}

\textbf{subroutine: hothalo\_angular\_momentum\_cold\_rate}

Description: Accept a rate set for the \texttt{angular MomentumCold} property of the \texttt{hotHalo} component class. Trigger an interrupt to create the component.

Code lines: 15

Contained by: module \texttt{galacticus\_nodes}

Modules used: \texttt{galacticus\_error}

\textbf{function: hothalo\_angular\_momentum\_isgettable}

Description: Returns true if the \texttt{angular Momentum} property is gettable for the \texttt{hotHalo} component class.

Code lines: 6

Contained by: module \texttt{galacticus\_nodes}

\textbf{subroutine: hothalo\_angular\_momentum\_rate}

Description: Accept a rate set for the \texttt{angular Momentum} property of the \texttt{hotHalo} component class. Trigger an interrupt to create the component.

Code lines: 15

Contained by: module \texttt{galacticus\_nodes}

Modules used: \texttt{galacticus\_error}

\textbf{function: hothalo\_chemicals}

Description: Returns the default value for the \texttt{chemicals} property for the \texttt{hotHalo} component class.

Code lines: 8

Contained by: module \texttt{galacticus\_nodes}

\textbf{function: hothalo\_chemicals\_attributematch}

Description: Return a text list of component implementations in the \texttt{hotHalo} class that have the desired attributes for the \texttt{chemicals} property.

Code lines: 25

Contained by: module \texttt{galacticus\_nodes}

Modules used: \texttt{iso\_varying\_string}

\textbf{function: hothalo\_chemicals\_isgettable}
19. Source Code Documentation

**Description:** Returns true if the chemicals property is gettable for the hotHalo component class.
**Code lines:** 6
**Contained by:** module galacticus_nodes

**subroutine: hothalochemicalsrat**

**Description:** Accept a rate set for the chemicals property of the hotHalo component class. Trigger an interrupt to create the component.
**Code lines:** 15
**Contained by:** module galacticus_nodes
**Modules used:** galacticus_error

**function: hothalocoldmodeabundancescoldcount**

**Description:** Return a count of the number of scalar properties in the abundancesCold property of the hotHaloColdMode component implementation.
**Code lines:** 7
**Contained by:** module galacticus_nodes

**function: hothalocoldmodeabundancescoldget**

**Description:** Return the abundancesCold property of the HotHaloColdMode component implementation.
**Code lines:** 8
**Contained by:** module galacticus_nodes

**subroutine: hothalocoldmodeabundancescoldrate**

**Description:** Accumulate to the abundancesCold property rate of change of the HotHaloColdMode component implementation.
**Code lines:** 10
**Contained by:** module galacticus_nodes

**subroutine: hothalocoldmodeabundancescoldscale**

**Description:** Set the abundancesCold property scale of the HotHaloColdMode component implementation.
**Code lines:** 8
**Contained by:** module galacticus_nodes

**subroutine: hothalocoldmodeabundancescoldset**

**Description:** Set the abundancesCold property of the HotHaloColdMode component implementation.
**Code lines:** 11
**Contained by:** module galacticus_nodes
**Modules used:** memory_management

**function: hothalocoldmodeangularmomentumcoldcount**

**Description:** Return a count of the number of scalar properties in the angularMomentumCold property of the hotHaloColdMode component implementation.
**Code lines:** 7
**Contained by:** module galacticus_nodes

**function: hothalocoldmodeangularmomentumcoldget**

**Description:** Return the angularMomentumCold property of the HotHaloColdMode component implementation.
**Code lines:** 8
**Contained by:** module galacticus_nodes
subroutine: hothalocoldmodeangularmomentumcoldrate
Description: Accumulate to the angularMomentumCold property rate of change of the HotHaloColdMode component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: hothalocoldmodeangularmomentumcoldscale
Description: Set the angularMomentumCold property scale of the HotHaloColdMode component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothalocoldmodeangularmomentumcoldset
Description: Set the angularMomentumCold property of the HotHaloColdMode component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: hothalocoldmodemasscoldcount
Description: Return a count of the number of scalar properties in the massCold property of the HotHaloColdMode component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: hothalocoldmodemasscoldget
Description: Return the massCold property of the HotHaloColdMode component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothalocoldmodemasscoldrate
Description: Accumulate to the massCold property rate of change of the HotHaloColdMode component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: hothalocoldmodemasscoldscale
Description: Set the massCold property scale of the HotHaloColdMode component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothalocoldmodemasscoldset
Description: Set the massCold property of the HotHaloColdMode component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: hothalocoldmodeouterradiussgrowthrate
Description: Call the deferred function for the outerRadiusGrowthRate method of the HotHaloColdMode component.
19. Source Code Documentation

subroutine: hothalocoldmodeouterradiusgrowthratedeferredfunctionset
Description: Set the function to be used for the outerRadiusGrowthRate method of the HotHaloColdMode component.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalocoldmodeouterradiusgrowthratedfrrdfnctnisset
Description: Return true if the deferred function for the outerRadiusGrowthRate method of the HotHaloColdMode component has been set.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: hothalocoldmodeoutflowreturn
Description: Call the deferred function for the outflowReturn method of the HotHaloColdMode component.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: hothalocoldmodeoutflowreturndfrrdfnctnisset
Description: Set the function to be used for the outflowReturn method of the HotHaloColdMode component.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalocoldmodeoutflowreturndfrrdfnctnisset
Description: Return true if the deferred function for the outflowReturn method of the HotHaloColdMode component has been set.
Code lines: 5
Contained by: module galacticus_nodes

function: hothalocountlinked
Description: Returns the number of hotHalo components in self.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: hothalocreatebyinterrupt
Description: Create the hotHalo component of self via an interrupt.
Code lines: 12
Contained by: module galacticus_nodes

subroutine: hothalocreatelinked
Description: Create the hotHalo component of self.
Code lines: 29
Contained by: module galacticus_nodes

Modules used: galacticus_display iso_varying_string
**19.1. Program units**

**string_handling**

**subroutine**: hothalodestroylinked  
*Description*: Destroy the hotHalo component of self.  
*Code lines*: 13  
*Contained by*: module galacticus_nodes

**function**: hothaloget  
*Description*: Returns the hotHalo component of self.  
*Code lines*: 27  
*Contained by*: module galacticus_nodes  
*Modules used*: galacticus_error

**function**: hothaloheatsource  
*Description*: Returns the default value for the heatSource property for the hotHalo component class.  
*Code lines*: 8  
*Contained by*: module galacticus_nodes

**function**: hothaloheatsourceattributematch  
*Description*: Return a text list of component implementations in the hotHalo class that have the desired attributes for the heatSource property  
*Code lines*: 27  
*Contained by*: module galacticus_nodes  
*Modules used*: iso_varying_string

**function**: hothaloheatsourceisgettable  
*Description*: Returns true if the heatSource property is gettable for the hotHalo component class.  
*Code lines*: 6  
*Contained by*: module galacticus_nodes

**function**: hothalohothalocoolingabundances  
*Description*: Returns the default value for the hotHaloCoolingAbundances property for the hotHalo component class.  
*Code lines*: 8  
*Contained by*: module galacticus_nodes

**function**: hothalohothalocoolingabundancesattributematch  
*Description*: Return a text list of component implementations in the hotHalo class that have the desired attributes for the hotHaloCoolingAbundances property  
*Code lines*: 27  
*Contained by*: module galacticus_nodes  
*Modules used*: iso_varying_string

**function**: hothalohothalocoolingabundancesisgettable  
*Description*: Returns true if the hotHaloCoolingAbundances property is gettable for the hotHalo component class.  
*Code lines*: 6  
*Contained by*: module galacticus_nodes

**subroutine**: hothalohothalocoolingabundancesratefunction
19. Source Code Documentation

**Description:** Set the function to be used for rate of the hotHaloCoolingAbundances property of the hotHalo component class.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**function: hotspot_halo_halocoolingabundancesrateisattached**

**Description:** Return true if the deferred function used to rate the hotHaloCoolingAbundances property of the hotHalo component class has been attached.

**Code lines:** 5

**Contained by:** module galacticus_nodes

**function: hotspot_halo_halocoolingangularmomentum**

**Description:** Returns the default value for the hotHaloCoolingAngularMomentum property for the hotHalo component class.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**function: hotspot_halo_halocoolingangularmomentumattributematch**

**Description:** Return a text list of component implementations in the hotHalo class that have the desired attributes for the hotHaloCoolingAngularMomentum property

**Code lines:** 27

**Contained by:** module galacticus_nodes

**Modules used:** iso_varying_string

**function: hotspot_halo_halocoolingangularmomentumisgettable**

**Description:** Returns true if the hotHaloCoolingAngularMomentum property is gettable for the hotHalo component class.

**Code lines:** 6

**Contained by:** module galacticus_nodes

**subroutine: hotspot_halo_halocoolingangularmomentumratefunction**

**Description:** Set the function to be used for rate of the hotHaloCoolingAngularMomentum property of the hotHalo component class.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**function: hotspot_halo_halocoolingangularmomentumrateisattached**

**Description:** Return true if the deferred function used to rate the hotHaloCoolingAngularMomentum property of the hotHalo component class has been attached.

**Code lines:** 5

**Contained by:** module galacticus_nodes

**function: hotspot_halo_halocoolingmass**

**Description:** Returns the default value for the hotHaloCoolingMass property for the hotHalo component class.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**function: hotspot_halo_halocoolingmassattributematch**

**Description:** Return a text list of component implementations in the hotHalo class that have the desired attributes for the hotHaloCoolingMass property
19.1. Program units

Code lines: 38
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: hothalohothalocoolingmassisgettable
Description: Returns true if the hotHaloCoolingMass property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: hothalohothalocoolingmassratefunction
Description: Set the function to be used for rate of the hotHaloCoolingMass property of the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalohothalocoolingmassrateisattached
Description: Return true if the deferred function used to rate the hotHaloCoolingMass property of the hotHalo component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes

function: hothaloisinitialized
Description: Returns the default value for the isInitialized property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothaloisinitializedattributematch
Description: Return a text list of component implementations in the hotHalo class that have the desired attributes for the isInitialized property.
Code lines: 27
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: hothaloisinitializedisgettable
Description: Returns true if the isInitialized property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

function: hothalomass
Description: Returns the default value for the mass property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalomassattributematch
Description: Return a text list of component implementations in the hotHalo class that have the desired attributes for the mass property.
Code lines: 34
Contained by: module galacticus_nodes
Modules used: iso_varying_string
function: hothalomasscold
  Description: Returns the default value for the massCold property for the hotHalo component class.
  Code lines: 8
  Contained by: module galacticus_nodes

function: hothalomasscoldattributematch
  Description: Return a text list of component implementations in the hotHalo class that have the desired attributes for the massCold property.
  Code lines: 25
  Contained by: module galacticus_nodes
  Modules used: iso_varying_string

function: hothalomasscoldisgettable
  Description: Returns true if the massCold property is gettable for the hotHalo component class.
  Code lines: 6
  Contained by: module galacticus_nodes

subroutine: hothalomasscoldrate
  Description: Accept a rate set for the massCold property of the hotHalo component class. Trigger an interrupt to create the component.
  Code lines: 15
  Contained by: module galacticus_nodes
  Modules used: galacticus_error

function: hothalomassisgettable
  Description: Returns true if the mass property is gettable for the hotHalo component class.
  Code lines: 6
  Contained by: module galacticus_nodes

subroutine: hothalomassrate
  Description: Accept a rate set for the mass property of the hotHalo component class. Trigger an interrupt to create the component.
  Code lines: 15
  Contained by: module galacticus_nodes
  Modules used: galacticus_error

function: hothalomasssink
  Description: Returns the default value for the massSink property for the hotHalo component class.
  Code lines: 8
  Contained by: module galacticus_nodes

function: hothalomasssinkattributematch
  Description: Return a text list of component implementations in the hotHalo class that have the desired attributes for the massSink property.
  Code lines: 27
  Contained by: module galacticus_nodes
  Modules used: iso_varying_string

function: hothalomasssinkisgettable
19.1. Program units

**Description:** Returns true if the massSink property is gettable for the hotHalo component class.

**Code lines:** 6

**Contained by:** module galacticus_nodes

**Function:** hothalonullbindingabundances0inout

**Description:** A null get function for rank 0 type abundances.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**Function:** hothalonullbindingchemicalabundances0inout

**Description:** A null get function for rank 0 type chemical abundances.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**Function:** hothalonullbindingdouble0inout

**Description:** A null get function for rank 0 double precisions.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**Function:** hothalonullbindinginteger0in

**Description:** A null get function for rank 0 integers.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**Function:** hothalonullbindinglogical0inout

**Description:** A null get function for rank 0 logicals.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**Subroutine:** hothalonullbindingrateabundances0inout

**Description:** A null rate function for rank 0 type abundances.

**Code lines:** 9

**Contained by:** module galacticus_nodes

**Subroutine:** hothalonullbindingratechemicalabundances0inout

**Description:** A null rate function for rank 0 type chemical abundances.

**Code lines:** 9

**Contained by:** module galacticus_nodes

**Subroutine:** hothalonullbindingratedouble0inout

**Description:** A null rate function for rank 0 double precisions.

**Code lines:** 9

**Contained by:** module galacticus_nodes

**Subroutine:** hothalonullbindingrateinteger0in

**Description:** A null rate function for rank 0 integers.

**Code lines:** 9

**Contained by:** module galacticus_nodes
subroutine: hothalonullbindingratelogical0inout
Description: A null rate function for rank 0 logicals.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: hothalonullbindingsetabundances0inout
Description: A null set function for rank 0 type abundances.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: hothalonullbindingsetchemicalabundances0inout
Description: A null set function for rank 0 type chemical abundances.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: hothalonullbindingsetdouble0inout
Description: A null set function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: hothalonullbindingsetinteger0in
Description: A null set function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: hothalonullbindingsetlogical0inout
Description: A null set function for rank 0 logicals.
Code lines: 7
Contained by: module galacticus_nodes

function: hothaloouterradius
Description: Returns the default value for the outerRadius property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothaloouterradiusattributematch
Description: Return a text list of component implementations in the hotHalo class that have the desired attributes for the outerRadius property.
Code lines: 36
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: hothaloouterradiusisgettable
Description: Returns true if the outerRadius property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

function: hothalooutflowedabundances
19.1. Program units

**Description:** Returns the default value for the `outflowedAbundances` property for the `hotHalo` component class.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**function: hothalooutflowedabundancesattributematch**

**Description:** Return a text list of component implementations in the `hotHalo` class that have the desired attributes for the `outflowedAbundances` property.

**Code lines:** 25

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function: hothalooutflowedabundancesisgettable**

**Description:** Returns true if the `outflowedAbundances` property is gettable for the `hotHalo` component class.

**Code lines:** 6

**Contained by:** module `galacticus_nodes`

**function: hothalooutflowedangularmomentum**

**Description:** Returns the default value for the `outflowedAngularMomentum` property for the `hotHalo` component class.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**function: hothalooutflowedangularmomentumattributematch**

**Description:** Return a text list of component implementations in the `hotHalo` class that have the desired attributes for the `outflowedAngularMomentum` property.

**Code lines:** 25

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function: hothalooutflowedangularmomentumisgettable**

**Description:** Returns true if the `outflowedAngularMomentum` property is gettable for the `hotHalo` component class.

**Code lines:** 6

**Contained by:** module `galacticus_nodes`

**function: hothalooutflowedmass**

**Description:** Returns the default value for the `outflowedMass` property for the `hotHalo` component class.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**function: hothalooutflowedmassattributematch**

**Description:** Return a text list of component implementations in the `hotHalo` class that have the desired attributes for the `outflowedMass` property.

**Code lines:** 25

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function: hothalooutflowedmassisgettable**
19. Source Code Documentation

**Description:** Returns true if the `outflowedMass` property is gettable for the `hotHalo` component class.
**Code lines:** 6
**Contained by:** module `galacticus_nodes`

**function:** `hothalayoutflowingabundances`
**Description:** Returns the default value for the `outflowingAbundances` property for the `hotHalo` component class.
**Code lines:** 8
**Contained by:** module `galacticus_nodes`

**function:** `hothalayoutflowingabundancesattributematch`
**Description:** Return a text list of component implementations in the `hotHalo` class that have the desired attributes for the `outflowingAbundances` property
**Code lines:** 27
**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function:** `hothalayoutflowingabundancesisgettable`
**Description:** Returns true if the `outflowingAbundances` property is gettable for the `hotHalo` component class.
**Code lines:** 6
**Contained by:** module `galacticus_nodes`

**function:** `hothalayoutflowingangularmomentum`
**Description:** Returns the default value for the `outflowingAngularMomentum` property for the `hotHalo` component class.
**Code lines:** 8
**Contained by:** module `galacticus_nodes`

**function:** `hothalayoutflowingangularmomentumattributematch`
**Description:** Return a text list of component implementations in the `hotHalo` class that have the desired attributes for the `outflowingAngularMomentum` property
**Code lines:** 27
**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function:** `hothalayoutflowingangularmomentumisgettable`
**Description:** Returns true if the `outflowingAngularMomentum` property is gettable for the `hotHalo` component class.
**Code lines:** 6
**Contained by:** module `galacticus_nodes`

**function:** `hothalayoutflowingmass`
**Description:** Returns the default value for the `outflowingMass` property for the `hotHalo` component class.
**Code lines:** 8
**Contained by:** module `galacticus_nodes`

**function:** `hothalayoutflowingmassattributematch`
**Description:** Return a text list of component implementations in the `hotHalo` class that have the desired attributes for the `outflowingMass` property
19.1. Program units

Code lines: 38
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: hothalooutflowingmassisgettable
Description: Returns true if the outflowingMass property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

function: hothalooutflowtrackingouterradiusgrowthrate
Description: Call the deferred function for the outerRadiusGrowthRate method of the HotHaloOutflowTracking component.
Code lines: 14
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: hothalooutflowtrackingouterradiusgrowthratedeferredfunctionset
Description: Set the function to be used for the outerRadiusGrowthRate method of the HotHaloOutflowTracking component.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalooutflowtrackingouterradiusgrowthratedfrrdfnctnisset
Description: Return true if the deferred function for the outerRadiusGrowthRate method of the HotHaloOutflowTracking component has been set.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: hothalooutflowtrackingoutflowreturn
Description: Call the deferred function for the outflowReturn method of the HotHaloOutflowTracking component.
Code lines: 16
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: hothalooutflowtrackingoutflowreturndeferredfunctionset
Description: Set the function to be used for the outflowReturn method of the HotHaloOutflowTracking component.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalooutflowtrackingoutflowreturndfrrdfnctnisset
Description: Return true if the deferred function for the outflowReturn method of the HotHaloOutflowTracking component has been set.
Code lines: 5
Contained by: module galacticus_nodes

function: hothalooutflowtrackingtrackedoutflowabundancescount
Description: Return a count of the number of scalar properties in the trackedOutflowAbundances property of the HotHaloOutflowTracking component implementation.
19. Source Code Documentation

function: hothalooutflowtrackingtrackedoutflowabundancesget
Description: Return the trackedOutflowAbundances property of the HotHaloOutflowTracking component implementation.

subroutine: hothalooutflowtrackingtrackedoutflowabundancesrate
Description: Accumulate to the trackedOutflowAbundances property rate of change of the HotHaloOutflowTracking component implementation.

subroutine: hothalooutflowtrackingtrackedoutflowabundancesscale
Description: Set the trackedOutflowAbundances property scale of the HotHaloOutflowTracking component implementation.

subroutine: hothalooutflowtrackingtrackedoutflowabundancesset
Description: Set the trackedOutflowAbundances property of the HotHaloOutflowTracking component implementation.

function: hothalooutflowtrackingtrackedoutflowmasscount
Description: Return a count of the number of scalar properties in the trackedOutflowMass property of the hotHaloOutflowTracking component implementation.

function: hothalooutflowtrackingtrackedoutflowmassget
Description: Return the trackedOutflowMass property of the HotHaloOutflowTracking component implementation.

subroutine: hothalooutflowtrackingtrackedoutflowmassrate
Description: Accumulate to the trackedOutflowMass property rate of change of the HotHaloOutflowTracking component implementation.

subroutine: hothalooutflowtrackingtrackedoutflowmassscale
Description: Set the trackedOutflowMass property scale of the HotHaloOutflowTracking component implementation.
subroutine: hothalooutflowtrackingtrackedoutflowmassset
Description: Set the trackedOutflowMass property of the HotHaloOutflowTracking component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: hothalostandardabundancescount
Description: Return a count of the number of scalar properties in the abundances property of the hotHaloStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: hothalostandardabundancesget
Description: Return the abundances property of the HotHaloStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothalostandardabundancesrate
Description: Accumulate to the abundances property rate of change of the HotHaloStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: hothalostandardabundancesscale
Description: Set the abundances property scale of the HotHaloStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothalostandardabundancesset
Description: Set the abundances property of the HotHaloStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: hothalostandardangularmomentumcount
Description: Return a count of the number of scalar properties in the angularMomentum property of the hotHaloStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: hothalostandardangularmomentumget
Description: Return the angularMomentum property of the HotHaloStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothalostandardangularmomentumrate
Description: Accumulate to the angularMomentum property rate of change of the HotHaloStandard component implementation.
Code lines: 10
19. Source Code Documentation

Contained by: module galacticus_nodes

subroutine: hothalostandardangularmomentumscale
Description: Set the angularMomentum property scale of the HotHaloStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothalostandardangularmomentumset
Description: Set the angularMomentum property of the HotHaloStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: hothalostandardchemicalscount
Description: Return a count of the number of scalar properties in the chemicals property of the HotHaloStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: hothalostandardchemicalsget
Description: Return the chemicals property of the HotHaloStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothalostandardchemicalsrate
Description: Accumulate to the chemicals property rate of change of the HotHaloStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: hothalostandardchemicalsscale
Description: Set the chemicals property scale of the HotHaloStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothalostandardchemicalsset
Description: Set the chemicals property of the HotHaloStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: hothalostandardcreatefunctionset
Description: Set the create function for the HotHaloStandard component.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: hothalostandardheatsourcerate
Description: Set the rate of the heatSource property of the HotHaloStandard component using a deferred function.
Code lines: 10
19.1. Program units

**Contained by:** module galacticus_nodes

**subroutine:** hothalostandardheatsourceratefunction
- **Description:** Set the function to be used for rate of the heatSource property of the HotHaloStandard component class.
- **Code lines:** 8
- **Contained by:** module galacticus_nodes

**function:** hothalostandardheatsourcerateisattached
- **Description:** Return true if the deferred function used to rate the heatSource property of the HotHaloStandard component class has been attached.
- **Code lines:** 5
- **Contained by:** module galacticus_nodes

**subroutine:** hothalostandardhothalocoolingabundancesrate
- **Description:** Set the rate of the hotHaloCoolingAbundances property of the HotHaloStandard component using a deferred function.
- **Code lines:** 10
- **Contained by:** module galacticus_nodes

**subroutine:** hothalostandardhothalocoolingangularmomentumrate
- **Description:** Set the rate of the hotHaloCoolingAngularMomentum property of the HotHaloStandard component using a deferred function.
- **Code lines:** 10
- **Contained by:** module galacticus_nodes

**subroutine:** hothalostandardhothalocoolingmassrate
- **Description:** Set the rate of the hotHaloCoolingMass property of the HotHaloStandard component using a deferred function.
- **Code lines:** 10
- **Contained by:** module galacticus_nodes

**function:** hothalostandardisinitializedget
- **Description:** Return the isInitialized property of the HotHaloStandard component implementation.
- **Code lines:** 8
- **Contained by:** module galacticus_nodes

**subroutine:** hothalostandardisinitializedset
- **Description:** Set the isInitialized property of the HotHaloStandard component implementation.
- **Code lines:** 9
- **Modules used:** memory_management

**function:** hothalostandardmasscount
- **Description:** Return a count of the number of scalar properties in the mass property of the HotHaloStandard component implementation.
- **Code lines:** 7
- **Contained by:** module galacticus_nodes

**function:** hothalostandardmassget
**Description:** Return the mass property of the HotHaloStandard component implementation.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**subroutine: hothalostandardmassrate**

**Description:** Accumulate to the mass property rate of change of the HotHaloStandard component implementation.

**Code lines:** 10

**Contained by:** module galacticus_nodes

**subroutine: hothalostandardmassscale**

**Description:** Set the mass property scale of the HotHaloStandard component implementation.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**subroutine: hothalostandardmassset**

**Description:** Set the mass property of the HotHaloStandard component implementation.

**Code lines:** 11

**Contained by:** module galacticus_nodes

**Modules used:** memory_management

**subroutine: hothalostandardmasssinkrate**

**Description:** Set the rate of the massSink property of the HotHaloStandard component using a deferred function.

**Code lines:** 10

**Contained by:** module galacticus_nodes

**subroutine: hothalostandardmasssinkratefunction**

**Description:** Set the function to be used for rate of the massSink property of the HotHaloStandard component class.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**function: hothalostandardmasssinkrateisattached**

**Description:** Return true if the deferred function used to rate the massSink property of the HotHaloStandard component class has been attached.

**Code lines:** 5

**Contained by:** module galacticus_nodes

**function: hothalostandardouterradiuscount**

**Description:** Return a count of the number of scalar properties in the outerRadius property of the HotHaloStandard component implementation.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**function: hothalostandardouterradiusget**

**Description:** Get the value of the outerRadius property of the HotHaloStandard component using a deferred function.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**subroutine: hothalostandardouterradiusgetfunction**
19.1. Program units

Description: Set the function to be used for get of the outerRadius property of the HotHaloStandard component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalostandardouterradiusgetisattached
Description: Return true if the deferred function used to get the outerRadius property of the HotHaloStandard component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes

function: hothalostandardouterradiusgetvalue
Description: Return the outerRadius property of the HotHaloStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalostandardouterradiusgrowthrate
Description: Call the deferred function for the outerRadiusGrowthRate method of the HotHaloStandard component.
Code lines: 14
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: hothalostandardouterradiusgrowthratedeferredfunctionset
Description: Set the function to be used for the outerRadiusGrowthRate method of the HotHaloStandard component.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalostandardouterradiusgrowthratedfrrdfnctnisset
Description: Return true if the deferred function for the outerRadiusGrowthRate method of the HotHaloStandard component has been set.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: hothalostandardouterradiusrate
Description: Accumulate to the outerRadius property rate of change of the HotHaloStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: hothalostandardouterradiusscale
Description: Set the outerRadius property scale of the HotHaloStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothalostandardouterradiusset
Description: Set the outerRadius property of the HotHaloStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
19. Source Code Documentation

**Modules used:** memory_management

**function:** hothalostandardoutflowedabundancescount

*Description:* Return a count of the number of scalar properties in the outflowedAbundances property of the hotHaloStandard component implementation.

*Code lines:* 7
*Contained by:* module galacticus_nodes

**function:** hothalostandardoutflowedabundancesget

*Description:* Return the outflowedAbundances property of the HotHaloStandard component implementation.

*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine:** hothalostandardoutflowedabundancesrate

*Description:* Accumulate to the outflowedAbundances property rate of change of the HotHaloStandard component implementation.

*Code lines:* 10
*Contained by:* module galacticus_nodes

**subroutine:** hothalostandardoutflowedabundancesscale

*Description:* Set the outflowedAbundances property scale of the HotHaloStandard component implementation.

*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine:** hothalostandardoutflowedabundancesset

*Description:* Set the outflowedAbundances property of the HotHaloStandard component implementation.

*Code lines:* 11
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**function:** hothalostandardoutflowedangularmomentumcount

*Description:* Return a count of the number of scalar properties in the outflowedAngularMomentum property of the hotHaloStandard component implementation.

*Code lines:* 7
*Contained by:* module galacticus_nodes

**function:** hothalostandardoutflowedangularmomentumget

*Description:* Return the outflowedAngularMomentum property of the HotHaloStandard component implementation.

*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine:** hothalostandardoutflowedangularmomentumrate

*Description:* Accumulate to the outflowedAngularMomentum property rate of change of the HotHaloStandard component implementation.

*Code lines:* 10
*Contained by:* module galacticus_nodes

**subroutine:** hothalostandardoutflowedangularmomentumscale
19.1. Program units

Description: Set the `outflowedAngularMomentum` property scale of the `HotHaloStandard` component implementation.

Code lines: 8

Contained by: module `galacticus_nodes`

**subroutine: hothalostandardoutflowedangularmomentumset**

Description: Set the `outflowedAngularMomentum` property of the `HotHaloStandard` component implementation.

Code lines: 11

Contained by: module `galacticus_nodes`

Modules used: `memory_management`

**function: hothalostandardoutflowedmasscount**

Description: Return a count of the number of scalar properties in the `outflowedMass` property of the `hotHaloStandard` component implementation.

Code lines: 7

Contained by: module `galacticus_nodes`

**function: hothalostandardoutflowedmassget**

Description: Return the `outflowedMass` property of the `HotHaloStandard` component implementation.

Code lines: 8

Contained by: module `galacticus_nodes`

**subroutine: hothalostandardoutflowedmassrate**

Description: Accumulate to the `outflowedMass` property rate of change of the `HotHaloStandard` component implementation.

Code lines: 10

Contained by: module `galacticus_nodes`

**subroutine: hothalostandardoutflowedmassscale**

Description: Set the `outflowedMass` property scale of the `HotHaloStandard` component implementation.

Code lines: 8

Contained by: module `galacticus_nodes`

**subroutine: hothalostandardoutflowedmassset**

Description: Set the `outflowedMass` property of the `HotHaloStandard` component implementation.

Code lines: 11

Contained by: module `galacticus_nodes`

Modules used: `memory_management`

**subroutine: hothalostandardoutflowingabundancesrate**

Description: Set the rate of the `outflowingAbundances` property of the `HotHaloStandard` component using a deferred function.

Code lines: 10

Contained by: module `galacticus_nodes`

**subroutine: hothalostandardoutflowingabundancesratefunction**

Description: Set the function to be used for rate of the `outflowingAbundances` property of the `HotHaloStandard` component class.

Code lines: 8

Contained by: module `galacticus_nodes`
function: hothalostandardoutflowingabundancesrateisattached
Description: Return true if the deferred function used to rate the `outflowingAbundances` property of the `HotHaloStandard` component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: hothalostandardoutflowingangularmomentumrate
Description: Set the rate of the `outflowingAngularMomentum` property of the `HotHaloStandard` component using a deferred function.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: hothalostandardoutflowingangularmomentumratefunction
Description: Set the function to be used for rate of the `outflowingAngularMomentum` property of the `HotHaloStandard` component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalostandardoutflowingangularmomentumrateisattached
Description: Return true if the deferred function used to rate the `outflowingAngularMomentum` property of the `HotHaloStandard` component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: hothalostandardoutflowingmassrate
Description: Set the rate of the `outflowingMass` property of the `HotHaloStandard` component using a deferred function.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: hothalostandardoutflowingmassratefunction
Description: Set the function to be used for rate of the `outflowingMass` property of the `HotHaloStandard` component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalostandardoutflowingmassrateisattached
Description: Return true if the deferred function used to rate the `outflowingMass` property of the `HotHaloStandard` component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: hothalostandardoutflowreturn
Description: Call the deferred function for the `outflowReturn` method of the `HotHaloStandard` component.
Code lines: 16
Contained by: module galacticus_nodes
Modules used: galacticus_error
19.1. Program units

**subroutine:** hothalostandardoutflowreturndeferredfunctionset
- **Description:** Set the function to be used for the `outflowReturn` method of the `HotHaloStandard` component.
- **Code lines:** 8
- **Contained by:** module `galacticus_nodes`

**function:** hothalostandardoutflowreturndfrrdfnctnisset
- **Description:** Return true if the deferred function for the `outflowReturn` method of the `HotHaloStandard` component has been set.
- **Code lines:** 5
- **Contained by:** module `galacticus_nodes`

**function:** hothalostandardstrippedabundancescount
- **Description:** Return a count of the number of scalar properties in the `strippedAbundances` property of the `hotHaloStandard` component implementation.
- **Code lines:** 7
- **Contained by:** module `galacticus_nodes`

**function:** hothalostandardstrippedabundancesget
- **Description:** Return the `strippedAbundances` property of the `HotHaloStandard` component implementation.
- **Code lines:** 8
- **Contained by:** module `galacticus_nodes`

**subroutine:** hothalostandardstrippedabundancesrate
- **Description:** Accumulate to the `strippedAbundances` property rate of change of the `HotHaloStandard` component implementation.
- **Code lines:** 10
- **Contained by:** module `galacticus_nodes`

**subroutine:** hothalostandardstrippedabundancesscale
- **Description:** Set the `strippedAbundances` property scale of the `HotHaloStandard` component implementation.
- **Code lines:** 8
- **Contained by:** module `galacticus_nodes`

**subroutine:** hothalostandardstrippedabundancesset
- **Description:** Set the `strippedAbundances` property of the `HotHaloStandard` component implementation.
- **Code lines:** 11
- **Contained by:** module `galacticus_nodes`
- **Modules used:** `memory_management`

**function:** hothalostandardstrippedmasscount
- **Description:** Return a count of the number of scalar properties in the `strippedMass` property of the `hotHaloStandard` component implementation.
- **Code lines:** 7
- **Contained by:** module `galacticus_nodes`

**function:** hothalostandardstrippedmassget
Description: Return the strippedMass property of the HotHaloStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothalostandardstrippedmassrate
Description: Accumulate to the strippedMass property rate of change of the HotHaloStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: hothalostandardstrippedmassscale
Description: Set the strippedMass property scale of the HotHaloStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothalostandardstrippedmassset
Description: Set the strippedMass property of the HotHaloStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: hothalostandardunaccretedmasscount
Description: Return a count of the number of scalar properties in the unaccretedMass property of the HotHaloStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: hothalostandardunaccretedmassget
Description: Return the unaccretedMass property of the HotHaloStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothalostandardunaccretedmassrate
Description: Accumulate to the unaccretedMass property rate of change of the HotHaloStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: hothalostandardunaccretedmassscale
Description: Set the unaccretedMass property scale of the HotHaloStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothalostandardunaccretedmassset
Description: Set the unaccretedMass property of the HotHaloStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management
function: hothalostrippedabundances
Description: Returns the default value for the strippedAbundances property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalostrippedabundancesattributematch
Description: Return a text list of component implementations in the hotHalo class that have the desired attributes for the strippedAbundances property.
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: hothalostrippedabundancesisgettable
Description: Returns true if the strippedAbundances property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

function: hothalotrackedoutflowabundances
Description: Returns the default value for the trackedOutflowAbundances property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalotrackedoutflowabundancesattributematch
Description: Return a text list of component implementations in the hotHalo class that have the desired attributes for the trackedOutflowAbundances property.
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: hothalotrackedoutflowabundancesisgettable
Description: Returns true if the trackedOutflowAbundances property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

function: hothalostrippedmass
Description: Returns the default value for the strippedMass property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalostrippedmassattributematch
Description: Return a text list of component implementations in the hotHalo class that have the desired attributes for the strippedMass property.
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: hothalostrippedmassisgettable
Description: Returns true if the strippedMass property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes
function: hothalotrackedoutflowabundancesisgettable
Description: Returns true if the trackedOutflowAbundances property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

function: hothalotrackedoutflowmass
Description: Returns the default value for the trackedOutflowMass property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalotrackedoutflowmassattributematch
Description: Return a text list of component implementations in the hotHalo class that have the desired attributes for the trackedOutflowMass property
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: hothalotrackedoutflowmassisgettable
Description: Returns true if the trackedOutflowMass property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

function: hothalounaccretedmass
Description: Returns the default value for the unaccretedMass property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalounaccretedmassattributematch
Description: Return a text list of component implementations in the hotHalo class that have the desired attributes for the unaccretedMass property
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: hothalounaccretedmassisgettable
Description: Returns true if the unaccretedMass property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: hothalounaccretedmassrate
Description: Accept a rate set for the unaccretedMass property of the hotHalo component class. Trigger an interrupt to create the component.
Code lines: 15
Contained by: module galacticus_nodes
Modules used: galacticus_error
19.1. Program units

subroutine: hothaloverysimplehothalocoolingmassrate
Description: Set the rate of the hotHaloCoolingMass property of the HotHaloVerySimple component using a deferred function.
Code lines: 10
Contained by: module galacticus_nodes

function: hothaloverysimplemasscount
Description: Return a count of the number of scalar properties in the mass property of the hotHaloVerySimple component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: hothaloverysimplemassget
Description: Return the mass property of the HotHaloVerySimple component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothaloverysimplemassrate
Description: Accumulate to the mass property rate of change of the HotHaloVerySimple component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: hothaloverysimplemassscale
Description: Set the mass property scale of the HotHaloVerySimple component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothaloverysimplemassset
Description: Set the mass property of the HotHaloVerySimple component implementation.
Code lines: 11
Contained by: module galacticus_nodes

Modules used: memory_management

function: hothaloverysimpleouterradiusget
Description: Get the value of the outerRadius property of the HotHaloVerySimple component using a deferred function.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothaloverysimpleouterradiusgetfunction
Description: Set the function to be used for get of the outerRadius property of the HotHaloVerySimple component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothaloverysimpleouterradiusgetisattached
Description: Return true if the deferred function used to get the outerRadius property of the HotHaloVerySimple component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes

**subroutine: hothaloverysimpleoutflowingmassrate**

*Description:* Set the rate of the `outflowingMass` property of the `HotHaloVerySimple` component using a deferred function.

Code lines: 10
Contained by: module galacticus_nodes

**subroutine: hothaloverysimpleoutflowingmassratefunction**

*Description:* Set the function to be used for rate of the `outflowingMass` property of the `HotHaloVerySimple` component class.

Code lines: 8
Contained by: module galacticus_nodes

**function: hothaloverysimpleoutflowingmassrateisattached**

*Description:* Return true if the deferred function used to rate the `outflowingMass` property of the `HotHaloVerySimple` component class has been attached.

Code lines: 5
Contained by: module galacticus_nodes

**function: indicesbranchtip**

*Description:* Returns the default value for the `branchTip` property for the `indices` component class.

Code lines: 8
Contained by: module galacticus_nodes

**function: indicesbranchtipattributematch**

*Description:* Return a text list of component implementations in the `indices` class that have the desired attributes for the `branchTip` property.

Code lines: 27
Contained by: module galacticus_nodes

**Modules used:** iso_varying_string

**function: indicesbranchtipisgettable**

*Description:* Returns true if the `branchTip` property is gettable for the `indices` component class.

Code lines: 6
Contained by: module galacticus_nodes

**function: indicescountlinked**

*Description:* Returns the number of `indices` components in `self`.

Code lines: 19
Contained by: module galacticus_nodes

**subroutine: indicescreatebyinterrupt**

*Description:* Create the `indices` component of `self` via an interrupt.

Code lines: 8
Contained by: module galacticus_nodes

**subroutine: indicescreatelinked**
19.1. Program units

*Description:* Create the `indices` component of `self`.
*Code lines:* 29
*Contained by:* module `galacticus_nodes`
*Modules used:* `galacticus_display` `iso_varying_string` `string_handling`

**subroutine:** `indicesdestroylinked`
*Description:* Destroy the `indices` component of `self`.
*Code lines:* 13
*Contained by:* module `galacticus_nodes`

**function:** `indicesget`
*Description:* Returns the `indices` component of `self`.
*Code lines:* 27
*Contained by:* module `galacticus_nodes`
*Modules used:* `galacticus_error`

**function:** `indicesnullbindinginteger0in`
*Description:* A null get function for rank 0 integers.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**function:** `indicesnullbindinglonginteger0inout`
*Description:* A null get function for rank 0 integer(kind=kind_int8)s.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**subroutine:** `indicesnullbindingrateinteger0in`
*Description:* A null rate function for rank 0 integers.
*Code lines:* 9
*Contained by:* module `galacticus_nodes`

**subroutine:** `indicesnullbindingratelonginteger0inout`
*Description:* A null rate function for rank 0 integer(kind=kind_int8)s.
*Code lines:* 9
*Contained by:* module `galacticus_nodes`

**subroutine:** `indicesnullbindingsetinteger0in`
*Description:* A null set function for rank 0 integers.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**subroutine:** `indicesnullbindingsetlonginteger0inout`
*Description:* A null set function for rank 0 integer(kind=kind_int8)s.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**function:** `indicesstandardbranchtipget`
*Description:* Return the `branchTip` property of the `IndicesStandard` component implementation.
subroutine: indicesstandardbranchtipset  
Description: Set the branchTip property of the IndicesStandard component implementation.

function: interoutputcountlinked  
Description: Returns the number of interOutput components in self.

subroutine: interoutputcreatebyinterrupt  
Description: Create the interOutput component of self via an interrupt.

subroutine: interoutputcreatelinked  
Description: Create the interOutput component of self.

subroutine: interoutputdestroylinked  
Description: Destroy the interOutput component of self.

function: interoutputdiskstarformationrate  
Description: Returns the default value for the diskStarFormationRate property for the interOutput component class.

function: interoutputdiskstarformationrateattributematch  
Description: Return a text list of component implementations in the interOutput class that have the desired attributes for the diskStarFormationRate property

function: interoutputdiskstarformationrateisgettable  
Description: Returns true if the diskStarFormationRate property is gettable for the interOutput component class.
subroutine: interoutputdiskstarformationraterate
Description: Accept a rate set for the diskStarFormationRate property of the interOutput component class. Trigger an interrupt to create the component.
Code lines: 15
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: interoutputget
Description: Returns the interOutput component of self.
Code lines: 27
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: interoutputnullbindingdouble0inout
Description: A null get function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

function: interoutputnullbindinginteger0in
Description: A null get function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: interoutputnullbindingratedouble0inout
Description: A null rate function for rank 0 double precisions.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: interoutputnullbindingrateinteger0in
Description: A null rate function for rank 0 integers.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: interoutputnullbindingsetdouble0inout
Description: A null set function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: interoutputnullbindingsetinteger0in
Description: A null set function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

function: interoutputspheroidstarformationrate
Description: Returns the default value for the spheroidStarFormationRate property for the interOutput component class.
Code lines: 12
Contained by: module galacticus_nodes
function: interoutputspheroidstarformationrateattributematch
Description: Return a text list of component implementations in the interOutput class that have the desired attributes for the spheroidStarFormationRate property
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: interoutputspheroidstarformationrateisgettable
Description: Returns true if the spheroidStarFormationRate property is gettable for the interOutput component class.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: interoutputspheroidstarformationraterate
Description: Accept a rate set for the spheroidStarFormationRate property of the interOutput component class. Trigger an interrupt to create the component.
Code lines: 15
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: interoutputstandarddiskstarformationratecount
Description: Return a count of the number of scalar properties in the diskStarFormationRate property of the InterOutputStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: interoutputstandarddiskstarformationrateget
Description: Return the diskStarFormationRate property of the InterOutputStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: interoutputstandarddiskstarformationraterate
Description: Accumulate to the diskStarFormationRate property rate of change of the InterOutputStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: interoutputstandarddiskstarformationratescale
Description: Set the diskStarFormationRate property scale of the InterOutputStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: interoutputstandarddiskstarformationrateset
Description: Set the diskStarFormationRate property of the InterOutputStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
19.1. Program units

Modules used: memory_management

function: interoutputstandardspheroidstarformationratecount
Description: Return a count of the number of scalar properties in the spheroidStarFormationRate property of the interOutputStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: interoutputstandardspheroidstarformationrateget
Description: Return the spheroidStarFormationRate property of the InterOutputStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: interoutputstandardspheroidstarformationraterate
Description: Accumulate to the spheroidStarFormationRate property rate of change of the InterOutputStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: interoutputstandardspheroidstarformationratescale
Description: Set the spheroidStarFormationRate property scale of the InterOutputStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: interoutputstandardspheroidstarformationrateset
Description: Set the spheroidStarFormationRate property of the InterOutputStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes

function: mapcomponentsdouble0
Description: Map a scalar double function over components with a specified reduction.
Code lines: 214
Contained by: module galacticus_nodes

subroutine: mapcomponentsvoid
Description: Map a void function over components.
Code lines: 98
Contained by: module galacticus_nodes

function: massflowstatisticscooledmass
Description: Returns the default value for the cooledMass property for the massFlowStatistics component class.
Code lines: 8
Contained by: module galacticus_nodes

function: massflowstatisticscooledmassattributematch
19. Source Code Documentation

**Description:** Return a text list of component implementations in the `massFlowStatistics` class that have the desired attributes for the `cooledMass` property

**Code lines:** 25

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function: massflowstatisticscooledmassisgettable**

**Description:** Returns true if the `cooledMass` property is gettable for the `massFlowStatistics` component class.

**Code lines:** 6

**Contained by:** module `galacticus_nodes`

**function: massflowstatisticscountlinked**

**Description:** Returns the number of `massFlowStatistics` components in `self`.

**Code lines:** 19

**Contained by:** module `galacticus_nodes`

**subroutine: massflowstatisticscreatebyinterrupt**

**Description:** Create the `massFlowStatistics` component of `self` via an interrupt.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**subroutine: massflowstatisticscreatelinked**

**Description:** Create the `massFlowStatistics` component of `self`.

**Code lines:** 29

**Contained by:** module `galacticus_nodes`

**Modules used:** `galacticus_display` `iso_varying_string` `string_handling`

**subroutine: massflowstatisticsdestroylinked**

**Description:** Destroy the `massFlowStatistics` component of `self`.

**Code lines:** 13

**Contained by:** module `galacticus_nodes`

**function: massflowstatisticsget**

**Description:** Returns the `massFlowStatistics` component of `self`.

**Code lines:** 27

**Contained by:** module `galacticus_nodes`

**Modules used:** `galacticus_error`

**function: massflowstatisticsnullbindingdouble0inout**

**Description:** A null get function for rank 0 double precisions.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**function: massflowstatisticsnullbindinginteger0in**

**Description:** A null get function for rank 0 integers.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`
19.1. Program units

subroutine: massflowstatisticsnullbindingratedouble0inout
Description: A null rate function for rank 0 double precisions.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: massflowstatisticsnullbindingrateinteger0in
Description: A null rate function for rank 0 integers.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: massflowstatisticsnullbindingsetdouble0inout
Description: A null set function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: massflowstatisticsnullbindingsetinteger0in
Description: A null set function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

function: massflowstatisticsstandardcooledmasscount
Description: Return a count of the number of scalar properties in the cooledMass property of the MassFlowStatisticsStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: massflowstatisticsstandardcooledmassget
Description: Return the cooledMass property of the MassFlowStatisticsStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: massflowstatisticsstandardcooledmassrate
Description: Accumulate to the cooledMass property rate of change of the MassFlowStatisticsStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: massflowstatisticsstandardcooledmassscale
Description: Set the cooledMass property scale of the MassFlowStatisticsStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: massflowstatisticsstandardcooledmassset
Description: Set the cooledMass property of the MassFlowStatisticsStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management
function: merger_tree_create_event
Description: Create a new event in a merger tree.
Code lines: 21
Contained by: module galacticus_nodes

subroutine: merger_tree_destroy
Description: Destroys the entire merger tree.
Code lines: 21
Contained by: module galacticus_nodes

subroutine: merger_tree_destroy_branch
Description: Destroy a branch of a tree which begins at thisNode.
Code lines: 35
Contained by: module galacticus_nodes

function: merger_tree_earliest_time
Description: Return the earliest time in a merger tree.
Code lines: 23
Contained by: module galacticus_nodes

function: merger_tree_earliest_time_evolving
Description: Return the earliest time in a merger tree.
Code lines: 23
Contained by: module galacticus_nodes

function: merger_tree_latest_time
Description: Return the latest time in a merger tree.
Code lines: 18
Contained by: module galacticus_nodes

function: merger_tree_node_get
Description: Return a pointer to a node in thisTree given the index of the node.
Code lines: 18
Contained by: module galacticus_nodes

subroutine: merger_tree_remove_event
Description: Removed an event from self.
Code lines: 27
Contained by: module galacticus_nodes

function: merger_tree_walk_descend_to_progenitors
Description: Descend to the deepest progenitor (satellites and children) of self.
Code lines: 18
Contained by: module galacticus_nodes

type: mergertree
Description: The merger tree object type.
Code lines: 71
19.1. Program units

Contained by: module galacticus_nodes

type: mergertreelist
Description: A class used for building linked lists of merger trees.
Code lines: 4
Contained by: module galacticus_nodes

function: mergingstatisticscountlinked
Description: Returns the number of mergingStatistics components in self.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: mergingstatisticscreatebyinterrupt
Description: Create the mergingStatistics component of self via an interrupt.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: mergingstatisticscreatelinked
Description: Create the mergingStatistics component of self.
Code lines: 29
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: mergingstatisticsdestroylinked
Description: Destroy the mergingStatistics component of self.
Code lines: 13
Contained by: module galacticus_nodes

function: mergingstatisticsgalaxymajormergertime
Description: Returns the default value for the galaxyMajorMergerTime property for the mergingStatistics component class.
Code lines: 12
Contained by: module galacticus_nodes

function: mergingstatisticsgalaxymajormergertimeattributematch
Description: Return a text list of component implementations in the mergingStatistics class that have the desired attributes for the galaxyMajorMergerTime property
Code lines: 27
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: mergingstatisticsgalaxymajormergertimeisgettable
Description: Returns true if the galaxyMajorMergerTime property is gettable for the mergingStatistics component class.
Code lines: 7
Contained by: module galacticus_nodes

function: mergingstatisticsget
19. Source Code Documentation

**Description:** Returns the `mergingStatistics` component of `self`.

**Code lines:** 27

**Contained by:** module `galacticus_nodes`

**Modules used:** `galacticus_error`

**function:** `mergingstatisticsnodeformationtime`

**Description:** Returns the default value for the `nodeFormationTime` property for the `mergingStatistics` component class.

**Code lines:** 12

**Contained by:** module `galacticus_nodes`

**function:** `mergingstatisticsnodeformationtimeattributematch`

**Description:** Return a text list of component implementations in the `mergingStatistics` class that have the desired attributes for the `nodeFormationTime` property

**Code lines:** 27

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function:** `mergingstatisticsnodeformationtimeisgettable`

**Description:** Returns true if the `nodeFormationTime` property is gettable for the `mergingStatistics` component class.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**function:** `mergingstatisticsnodehierarchylevel`

**Description:** Returns the default value for the `nodeHierarchyLevel` property for the `mergingStatistics` component class.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**function:** `mergingstatisticsnodehierarchylevelattributematch`

**Description:** Return a text list of component implementations in the `mergingStatistics` class that have the desired attributes for the `nodeHierarchyLevel` property

**Code lines:** 27

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function:** `mergingstatisticsnodehierarchylevelisgettable`

**Description:** Returns true if the `nodeHierarchyLevel` property is gettable for the `mergingStatistics` component class.

**Code lines:** 6

**Contained by:** module `galacticus_nodes`

**function:** `mergingstatisticsnodemajormergertime`

**Description:** Returns the default value for the `nodeMajorMergerTime` property for the `mergingStatistics` component class.

**Code lines:** 12

**Contained by:** module `galacticus_nodes`

**function:** `mergingstatisticsnodemajormergertimeattributematch`
19.1. Program units

**Description:** Return a text list of component implementations in the `mergingStatistics` class that have the desired attributes for the `nodeMajorMergerTime` property.

**Code lines:** 27

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function:** `mergingstatisticsnodemajormergertimeisgettable`

**Description:** Returns true if the `nodeMajorMergerTime` property is gettable for the `mergingStatistics` component class.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**function:** `mergingstatisticsnullbindingdouble0inout`

**Description:** A null get function for rank 0 double precisions.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**function:** `mergingstatisticsnullbindinginteger0in`

**Description:** A null get function for rank 0 integers.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**function:** `mergingstatisticsnullbindinginteger0inout`

**Description:** A null get function for rank 0 integers.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**function:** `mergingstatisticsnullbindinginteger1inout`

**Description:** A null get function for rank 1 integers.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**subroutine:** `mergingstatisticsnullbindingratedouble0inout`

**Description:** A null rate function for rank 0 double precisions.

**Code lines:** 9

**Contained by:** module `galacticus_nodes`

**subroutine:** `mergingstatisticsnullbindingrateinteger0in`

**Description:** A null rate function for rank 0 integers.

**Code lines:** 9

**Contained by:** module `galacticus_nodes`

**subroutine:** `mergingstatisticsnullbindingrateinteger0inout`

**Description:** A null rate function for rank 0 integers.

**Code lines:** 9

**Contained by:** module `galacticus_nodes`

**subroutine:** `mergingstatisticsnullbindingrateinteger1inout`

**Description:** A null rate function for rank 1 integers.
19. Source Code Documentation

subroutine: mergingstatisticsnullbindingsetdouble0inout
Description: A null set function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: mergingstatisticsnullbindingsetinteger0in
Description: A null set function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: mergingstatisticsnullbindingsetinteger0inout
Description: A null set function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: mergingstatisticsnullbindingsetinteger1inout
Description: A null set function for rank 1 integers.
Code lines: 7
Contained by: module galacticus_nodes

function: mergingstatisticsrecentmajormergercount
Description: Returns the default value for the recentMajorMergerCount property for the mergingStatistics component class.
Code lines: 14
Contained by: module galacticus_nodes
Modules used: galacticus_output_times

function: mergingstatisticsrecentmajormergercountattributematch
Description: Return a text list of component implementations in the mergingStatistics class that have the desired attributes for the recentMajorMergerCount property
Code lines: 27
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: mergingstatisticsrecentmajormergercountisgettable
Description: Returns true if the recentMajorMergerCount property is gettable for the mergingStatistics component class.
Code lines: 7
Contained by: module galacticus_nodes

function: mergingstatisticsrecentrecentmajormergercountget
Description: Return the recentMajorMergerCount property of the MergingStatisticsRecent component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: mergingstatisticsrecentrecentmajormergercountset
19.1. Program units

**Description:** Set the `recentMajorMergerCount` property of the `MergingStatisticsRecent` component implementation.

**Code lines:** 17

**Contained by:** module `galacticus_nodes`

**Modules used:** `memory_management`

**function:** `mergingstatisticsstandardgalaxymajormergertimeget`

**Description:** Return the `galaxyMajorMergerTime` property of the `MergingStatisticsStandard` component implementation.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**subroutine:** `mergingstatisticsstandardgalaxymajormergertimeset`

**Description:** Set the `galaxyMajorMergerTime` property of the `MergingStatisticsStandard` component implementation.

**Code lines:** 9

**Contained by:** module `galacticus_nodes`

**Modules used:** `memory_management`

**function:** `mergingstatisticsstandardnodeformationtimeget`

**Description:** Return the `nodeFormationTime` property of the `MergingStatisticsStandard` component implementation.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**subroutine:** `mergingstatisticsstandardnodeformationtimeset`

**Description:** Set the `nodeFormationTime` property of the `MergingStatisticsStandard` component implementation.

**Code lines:** 9

**Contained by:** module `galacticus_nodes`

**Modules used:** `memory_management`

**function:** `mergingstatisticsstandardnodehierarchylevelget`

**Description:** Return the `nodeHierarchyLevel` property of the `MergingStatisticsStandard` component implementation.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**subroutine:** `mergingstatisticsstandardnodehierarchylevelset`

**Description:** Set the `nodeHierarchyLevel` property of the `MergingStatisticsStandard` component implementation.

**Code lines:** 9

**Contained by:** module `galacticus_nodes`

**Modules used:** `memory_management`

**function:** `mergingstatisticsstandardnodemajormergertimeget`

**Description:** Return the `nodeMajorMergerTime` property of the `MergingStatisticsStandard` component implementation.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`
subroutine: mergingstatisticsstandardnode\textup{\texttt{\texttt{major}}mergertime}set
\textbf{Description:} Set the \texttt{\texttt{nodeMajorMergerTime}} property of the \texttt{MergingStatisticsStandard} component implementation.
\textbf{Code lines:} 9
\textbf{Contained by:} module \texttt{galacticus\_nodes}
\textbf{Modules used:} memory\_management

\textbf{function: nbodycountlinked}
\textbf{Description:} Returns the number of \texttt{nBody} components in \texttt{self}.
\textbf{Code lines:} 19
\textbf{Contained by:} module \texttt{galacticus\_nodes}

\textbf{subroutine: nbodycreatebyinterrupt}
\textbf{Description:} Create the \texttt{nBody} component of \texttt{self} via an interrupt.
\textbf{Code lines:} 8
\textbf{Contained by:} module \texttt{galacticus\_nodes}

\textbf{subroutine: nbodycreatelinked}
\textbf{Description:} Create the \texttt{nBody} component of \texttt{self}.
\textbf{Code lines:} 29
\textbf{Contained by:} module \texttt{galacticus\_nodes}
\textbf{Modules used:} galacticus\_display \texttt{\texttt{iso\_varying\_string}} \texttt{\texttt{string\_handling}}

\textbf{subroutine: nbodydestroylinked}
\textbf{Description:} Destroy the \texttt{nBody} component of \texttt{self}.
\textbf{Code lines:} 13
\textbf{Contained by:} module \texttt{galacticus\_nodes}

\textbf{function: nbodyget}
\textbf{Description:} Returns the \texttt{nBody} component of \texttt{self}.
\textbf{Code lines:} 27
\textbf{Contained by:} module \texttt{galacticus\_nodes}
\textbf{Modules used:} galacticus\_error

\textbf{function: nbodynullbindingdouble0inout}
\textbf{Description:} A null get function for rank 0 double precisions.
\textbf{Code lines:} 7
\textbf{Contained by:} module \texttt{galacticus\_nodes}

\textbf{function: nbodynullbindinginteger0in}
\textbf{Description:} A null get function for rank 0 integers.
\textbf{Code lines:} 7
\textbf{Contained by:} module \texttt{galacticus\_nodes}

\textbf{function: nbodynullbindinglonginteger0inout}
\textbf{Description:} A null get function for rank 0 integer\texttt{(kind=kind\_int8)s}.
\textbf{Code lines:} 7
19.1. Program units

```
Contained by: module galacticus_nodes

subroutine: nbodynullbindingratedouble0inout
  Description: A null rate function for rank 0 double precisions.
  Code lines: 9
  Contained by: module galacticus_nodes

subroutine: nbodynullbindingrateinteger0in
  Description: A null rate function for rank 0 integers.
  Code lines: 9
  Contained by: module galacticus_nodes

subroutine: nbodynullbindingratelonginteger0inout
  Description: A null rate function for rank 0 integer(kind=kind_int8)s.
  Code lines: 9
  Contained by: module galacticus_nodes

subroutine: nbodynullbindingsetdouble0inout
  Description: A null set function for rank 0 double precisions.
  Code lines: 7
  Contained by: module galacticus_nodes

subroutine: nbodynullbindingsetinteger0in
  Description: A null set function for rank 0 integers.
  Code lines: 7
  Contained by: module galacticus_nodes

subroutine: nbodynullbindingsetlonginteger0inout
  Description: A null set function for rank 0 integer(kind=kind_int8)s.
  Code lines: 7
  Contained by: module galacticus_nodes

function: nbodyparticlecount
  Description: Returns the default value for the particleCount property for the nBody component class.
  Code lines: 8
  Contained by: module galacticus_nodes

function: nbodyparticlecountattributematch
  Description: Return a text list of component implementations in the nBody class that have the desired attributes for the particleCount property
  Code lines: 27
  Contained by: module galacticus_nodes
  Modules used: iso_varying_string

function: nbodyparticlecountsgettable
  Description: Returns true if the particleCount property is gettable for the nBody component class.
  Code lines: 6
  Contained by: module galacticus_nodes
```
function: nbodystandardparticlecountget
Description: Return the particleCount property of the NBodyStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: nbodystandardparticlecountset
Description: Set the particleCount property of the NBodyStandard component implementation.
Code lines: 9
Contained by: module galacticus_nodes

function: nbodystandardvelocitydispersionget
Description: Return the velocityDispersion property of the NBodyStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: nbodystandardvelocitydispersionset
Description: Set the velocityDispersion property of the NBodyStandard component implementation.
Code lines: 9
Contained by: module galacticus_nodes

function: nbodystandardvelocitymaximumget

Description: Return the velocityMaximum property of the NBodyStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: nbodystandardvelocitymaximumset
Description: Set the velocityMaximum property of the NBodyStandard component implementation.
Code lines: 9
Contained by: module galacticus_nodes

function: nbodyvelocitydispersion
Description: Returns the default value for the velocityDispersion property for the nBody component class.
Code lines: 8
Contained by: module galacticus_nodes

function: nbodyvelocitydispersionattributematch
Description: Return a text list of component implementations in the nBody class that have the desired attributes for the velocityDispersion property
Code lines: 27
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: nbodyvelocitydispersionisgettable
19.1. Program units

**Description:** Returns true if the `velocityDispersion` property is gettable for the `nBody` component class.
**Code lines:** 6
**Contained by:** module `galacticus_nodes`

**function: nbodyvelocitymaximum**
**Description:** Returns the default value for the `velocityMaximum` property for the `nBody` component class.
**Code lines:** 8
**Contained by:** module `galacticus_nodes`

**function: nbodyvelocitymaximumattributematch**
**Description:** Return a text list of component implementations in the `nBody` class that have the desired attributes for the `velocityMaximum` property.
**Code lines:** 27
**Contained by:** module `galacticus_nodes`
**Modules used:** `iso_varying_string`

**function: nbodyvelocitymaximumisgettable**
**Description:** Returns true if the `velocityMaximum` property is gettable for the `nBody` component class.
**Code lines:** 6
**Contained by:** module `galacticus_nodes`

**subroutine: node_component_agestatistics_builder**
**Description:** Build a generic ageStatistics component.
**Code lines:** 10
**Contained by:** module `galacticus_nodes`
**Modules used:** `galacticus_error`

**subroutine: node_component_agestatistics_destroy**
**Description:** Destroys an ageStatistics component.
**Code lines:** 7
**Contained by:** module `galacticus_nodes`

**subroutine: node_component_agestatistics_dump**
**Description:** Dump the contents of a generic ageStatistics component.
**Code lines:** 10
**Contained by:** module `galacticus_nodes`
**Modules used:** `galacticus_display` `iso_varying_string`

**subroutine: node_component_agestatistics_initializor**
**Description:** Initialize a generic ageStatistics component.
**Code lines:** 8
**Contained by:** module `galacticus_nodes`
**Modules used:** `galacticus_error`

**subroutine: node_component_agestatistics_move**
**Description:** Move instances of the ageStatistics component, from one node to another.
**Code lines:** 67
**Contained by:** module `galacticus_nodes`
**Modules used:** `galacticus_error`
subroutine: node_component_agestatistics_output
Description: Output ptoperties for a ageStatistics component.
Code lines: 21
Contained by: module galacticus_nodes

subroutine: node_component_agestatistics_output_count
Description: Increment the count of properties to output for a generic ageStatistics component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_agestatistics_output_names
Description: Establish property names for a generic ageStatistics component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_agestatistics_remove
Description: Removes an instance of the ageStatistics component, shifting other instances to keep the array contiguous.
Code lines: 39
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: node_component_agestatistics_type
Description: Returns the type for the ageStatistics component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsnull_builder
Description: Build a null implementation of the ageStatistics component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_agestatisticsnull_count
Description: Return a count of the serialization of a null implementation of the ageStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsnull_deserialize_rates
Description: Serialize rates of a null implementation of the ageStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsnull_deserialize_scales
Description: Serialize scales of a null implementation of the ageStatistics component.
Code lines: 7
Contained by: module galacticus_nodes
19.1. Program units

subroutine: node_component_agestatisticsnull_deserialize_values
Description: Serialize values of a null implementation of the ageStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsnull_destroy
Description: Destroy a null implementation of the ageStatistics component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_agestatisticsnull_dump
Description: Dump the contents of a null implementation of the ageStatistics component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_agestatisticsnull_dump_raw
Description: Dump the contents of a null implementation of the ageStatistics component in binary.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsnull_dump_xml
Description: Dump the contents of a null implementation of the ageStatistics component to XML.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsnull_initializor
Description: Initialize a null implementation of the ageStatistics component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_agestatisticsnull_is_active
Description: Return true if the null implementation of the ageStatistics component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsnull_name_from_index
Description: Return the name of the property of given index for a null implementation of the ageStatistics component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_agestatisticsnull_ode_step_rates_init
Description: Initialize rates in a null implementation of the ageStatistics component for an ODE solver step.
19. Source Code Documentation

subroutine: node_component_agestatisticsnull_odestepscalesinit
Description: Initialize scales in a null implementation of the ageStatistics component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsnull_output_count
Description: Increment output property count for a null implementation of the ageStatistics component.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsnull_output_names
Description: Establish property names for a null implementation of the ageStatistics component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsnull_read_raw
Description: Read the contents of a null implementation of the ageStatistics component in binary.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_agestatisticsnull_serialize_rates
Description: Serialize rates of a null implementation of the ageStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsnull_serialize_scales
Description: Serialize scales of a null implementation of the ageStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsnull_serialize_values
Description: Serialize values of a null implementation of the ageStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_agestatisticsnull_type
Description: Returns the type for the null implementation of the ageStatistics component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsstandard_builder
Description: Build a standard implementation of the ageStatistics component.
Code lines: 37
Contained by: module galacticus_nodes
19.1. Program units

Modules used: fox_dom  memory_management  galacticus_error

function: node_component_agestatisticsstandard_count
Description: Return a count of the serialization of a standard implementation of the ageStatistics component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsstandard_deserialize_rates
Description: Serialize rates of a standard implementation of the ageStatistics component.
Code lines: 17
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsstandard_deserialize_scales
Description: Serialize scales of a standard implementation of the ageStatistics component.
Code lines: 17
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsstandard_deserialize_values
Description: Serialize values of a standard implementation of the ageStatistics component.
Code lines: 17
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsstandard_destroy
Description: Destroy a standard implementation of the ageStatistics component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_agestatisticsstandard_dump
Description: Dump the contents of a standard implementation of the ageStatistics component.
Code lines: 25
Contained by: module galacticus_nodes
Modules used: galacticus_display  iso_varying_string  string_handling

subroutine: node_component_agestatisticsstandard_dump_raw
Description: Dump the contents of a standard implementation of the ageStatistics component in binary.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsstandard_dump_xml
Description: Dump the contents of a standard implementation of the ageStatistics component to XML.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsstandard_initalizor
Description: Initialize a standard implementation of the ageStatistics component.
function: node_component_agestatisticsstandard_is_active
Description: Return true if the standard implementation of the ageStatistics component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsstandard_name_from_index
Description: Return the name of the property of given index for a standard implementation of the ageStatistics component.
Code lines: 29
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_agestatisticsstandard_ode_step_rates_init
Description: Initialize rates in a standard implementation of the ageStatistics component for an ODE solver step.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsstandard_ode_stepscalesinit
Description: Initialize scales in a standard implementation of the ageStatistics component for an ODE solver step.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsstandard_output_count
Description: Increment output property count for a standard implementation of the ageStatistics component.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsstandard_output_names
Description: Establish property names for a standard implementation of the ageStatistics component.
Code lines: 63
Contained by: module galacticus_nodes

subroutine: node_component_agestatisticsstandard_read_raw
Description: Read the contents of a standard implementation of the ageStatistics component in binary.
Code lines: 12
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_agestatisticsstandard_serialize_rates
Description: Serialize rates of a standard implementation of the ageStatistics component.
Code lines: 17
19.1. Program units

Contained by: module galacticus_nodes

**subroutine**: node_component_agestatisticsstandard_serialize_scales
Description: Serialize scales of a standard implementation of the ageStatistics component.
Code lines: 17
Contained by: module galacticus_nodes

**subroutine**: node_component_agestatisticsstandard_serialize_values
Description: Serialize values of a standard implementation of the ageStatistics component.
Code lines: 17
Contained by: module galacticus_nodes

**function**: node_component_agestatisticsstandard_type
Description: Returns the type for the standard implementation of the ageStatistics component.
Code lines: 8
Contained by: module galacticus_nodes

**subroutine**: node_component_assign
Description: Assign a node component to another node component.
Code lines: 358
Contained by: module galacticus_nodes

**subroutine**: node_component_basic_builder
Description: Build a generic basic component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_nodes galacticus_error

**subroutine**: node_component_basic_destroy
Description: Destroys a basic component.
Code lines: 7
Contained by: module galacticus_nodes

**subroutine**: node_component_basic_dump
Description: Dump the contents of a generic basic component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string

**subroutine**: node_component_basic_initializor
Description: Initialize a generic basic component.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: galacticus_error

**subroutine**: node_component_basic_move
Description: Move instances of the basic component, from one node to another.
Code lines: 95
Contained by: module galacticus_nodes

1473
19. Source Code Documentation

Modules used: galacticus_error

**subroutine: node_component_basic_output**
*Description:* Output properties for a basic component.
*Code lines:* 23
*Contained by:* module galacticus_nodes

**subroutine: node_component_basic_output_count**
*Description:* Increment the count of properties to output for a generic basic component.
*Code lines:* 13
*Contained by:* module galacticus_nodes

**subroutine: node_component_basic_output_names**
*Description:* Establish property names for a generic basic component.
*Code lines:* 15
*Contained by:* module galacticus_nodes

**subroutine: node_component_basic_remove**
*Description:* Removes an instance of the basic component, shifting other instances to keep the array contiguous.
*Code lines:* 55
*Contained by:* module galacticus_nodes
*Modules used:* galacticus_error

**function: node_component_basic_type**
*Description:* Returns the type for the basic component.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine: node_component_basicnonevolving_builder**
*Description:* Build a nonEvolving implementation of the basic component.
*Code lines:* 31
*Contained by:* module galacticus_nodes
*Modules used:* fox_dom galacticus_error memory_management

**function: node_component_basicnonevolving_count**
*Description:* Return a count of the serialization of a nonEvolving implementation of the basic component.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine: node_component_basicnonevolving_deserialize_rates**
*Description:* Serialize rates of a nonEvolving implementation of the basic component.
*Code lines:* 11
*Contained by:* module galacticus_nodes

**subroutine: node_component_basicnonevolving_deserialize_scales**
*Description:* Serialize scales of a nonEvolving implementation of the basic component.
*Code lines:* 11
19.1. Program units

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_basicnonevolving_deserialize_values`
*Description:* Serialize values of a nonEvolving implementation of the basic component.
*Code lines:* 11
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_basicnonevolving_destroy`
*Description:* Destroy a nonEvolving implementation of the basic component.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`
*Modules used:* `memory_management`

**subroutine:** `node_component_basicnonevolving_dump`
*Description:* Dump the contents of a nonEvolving implementation of the basic component.
*Code lines:* 22
*Contained by:* module `galacticus_nodes`
*Modules used:* `galacticus_display`  `iso_varying_string`  `string_handling`

**subroutine:** `node_component_basicnonevolving_dump_raw`
*Description:* Dump the contents of a nonEvolving implementation of the basic component in binary.
*Code lines:* 10
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_basicnonevolving_dump_xml`
*Description:* Dump the contents of a nonEvolving implementation of the basic component to XML.
*Code lines:* 12
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_basicnonevolving_initializor`
*Description:* Initialize a nonEvolving implementation of the basic component.
*Code lines:* 10
*Contained by:* module `galacticus_nodes`
*Modules used:* `memory_management`

**function:** `node_component_basicnonevolving_is_active`
*Description:* Return true if the nonEvolving implementation of the basic component is the active choice.
*Code lines:* 6
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_basicnonevolving_name_from_index`
*Description:* Return the name of the property of given index for a nonEvolving implementation of the basic component.
*Code lines:* 14
*Contained by:* module `galacticus_nodes`
*Modules used:* `iso_varying_string`

**subroutine:** `node_component_basicnonevolving_ode_step_rates_init`
Description: Initialize rates in a nonEvolving implementation of the basic component for an ODE solver step.

Code lines: 7

Contained by: module galacticus_nodes

subroutine: node_component_basicnonevolving_odestepsscalesinit

Description: Initialize scales in a nonEvolving implementation of the basic component for an ODE solver step.

Code lines: 6

Contained by: module galacticus_nodes

subroutine: node_component_basicnonevolving_output_count

Description: Increment output property count for a nonEvolving implementation of the basic component.

Code lines: 10

Contained by: module galacticus_nodes

subroutine: node_component_basicnonevolving_output_names

Description: Establish property names for a nonEvolving implementation of the basic component.

Code lines: 37

Contained by: module galacticus_nodes

subroutine: node_component_basicnonevolving_read_raw

Description: Read the contents of a nonEvolving implementation of the basic component in binary.

Code lines: 11

Contained by: module galacticus_nodes

Modules used: memory_management

subroutine: node_component_basicnonevolving_serialize_rates

Description: Serialize rates of a nonEvolving implementation of the basic component.

Code lines: 11

Contained by: module galacticus_nodes

subroutine: node_component_basicnonevolving_serialize_scales

Description: Serialize scales of a nonEvolving implementation of the basic component.

Code lines: 11

Contained by: module galacticus_nodes

subroutine: node_component_basicnonevolving_serialize_values

Description: Serialize values of a nonEvolving implementation of the basic component.

Code lines: 11

Contained by: module galacticus_nodes

function: node_component_basicnonevolving_type

Description: Returns the type for the nonEvolving implementation of the basic component.

Code lines: 8

Contained by: module galacticus_nodes

subroutine: node_component_basicnull_builder

Description: Build a null implementation of the basic component.
19.1. Program units

Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom
memory_management

function: node_component_basicnull_count
Description: Return a count of the serialization of a null implementation of the basic component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_basicnull_deserialize_rates
Description: Serialize rates of a null implementation of the basic component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_basicnull_deserialize_scales
Description: Serialize scales of a null implementation of the basic component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_basicnull_deserialize_values
Description: Serialize values of a null implementation of the basic component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_basicnull_destroy
Description: Destroy a null implementation of the basic component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_basicnull_dump
Description: Dump the contents of a null implementation of the basic component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: galacticus_display
iso_varying_string
string_handling

subroutine: node_component_basicnull_dump_raw
Description: Dump the contents of a null implementation of the basic component in binary.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_basicnull_dump_xml
Description: Dump the contents of a null implementation of the basic component to XML.
Code lines: 7
Contained by: module galacticus_nodes
subroutine: node_component_basicnull_initializor
Description: Initialize a null implementation of the basic component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_basicnull_is_active
Description: Return true if the null implementation of the basic component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_basicnull_name_from_index
Description: Return the name of the property of given index for a null implementation of the basic component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_basicnull_ode_step_rates_init
Description: Initialize rates in a null implementation of the basic component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_basicnull_odestepscalesinit
Description: Initialize scales in a null implementation of the basic component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_basicnull_output_count
Description: Increment output property count for a null implementation of the basic component.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_basicnull_output_names
Description: Establish property names for a null implementation of the basic component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_basicnull_read_raw
Description: Read the contents of a null implementation of the basic component in binary.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_basicnull_serialize_rates
Description: Serialize rates of a null implementation of the basic component.
Code lines: 7
Contained by: module galacticus_nodes
subroutine: node_component_basicnull_serialize_scales
Description: Serialize scales of a null implementation of the basic component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_basicnull_serialize_values
Description: Serialize values of a null implementation of the basic component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_basicnull_type
Description: Returns the type for the null implementation of the basic component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_basicstandard_builder
Description: Build a standard implementation of the basic component.
Code lines: 37
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_basicstandard_count
Description: Return a count of the serialization of a standard implementation of the basic component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_basicstandard_deserialize_rates
Description: Serialize rates of a standard implementation of the basic component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_basicstandard_deserialize_scales
Description: Serialize scales of a standard implementation of the basic component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_basicstandard_deserialize_values
Description: Serialize values of a standard implementation of the basic component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_basicstandard_destroy
Description: Destroy a standard implementation of the basic component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management
subroutine: node_component_basicstandard_dump
Description: Dump the contents of a standard implementation of the basic component.
Code lines: 25
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_basicstandard_dump_raw
Description: Dump the contents of a standard implementation of the basic component in binary.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_basicstandard_dump_xml
Description: Dump the contents of a standard implementation of the basic component to XML.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_basicstandard_initializor
Description: Initialize a standard implementation of the basic component.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_basicstandard_is_active
Description: Return true if the standard implementation of the basic component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_basicstandard_name_from_index
Description: Return the name of the property of given index for a standard implementation of the basic component.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_basicstandard_ode_step_rates_init
Description: Initialize rates in a standard implementation of the basic component for an ODE solver step.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_basicstandard_odestepscalesinit
Description: Initialize scales in a standard implementation of the basic component for an ODE solver step.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_basicstandard_output_count
Description: Increment output property count for a standard implementation of the basic component.
19.1. Program units

Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_basicstandard_output_names
Description: Establish property names for a standard implementation of the basic component.
Code lines: 37
Contained by: module galacticus_nodes

subroutine: node_component_basicstandard_read_raw
Description: Read the contents of a standard implementation of the basic component in binary.
Code lines: 12
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_basicstandard_serialize_rates
Description: Serialize rates of a standard implementation of the basic component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_basicstandard_serialize_scales
Description: Serialize scales of a standard implementation of the basic component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_basicstandard_serialize_values
Description: Serialize values of a standard implementation of the basic component.
Code lines: 13
Contained by: module galacticus_nodes

function: node_component_basicstandard_type
Description: Returns the type for the standard implementation of the basic component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_basicstandardtracking_builder
Description: Build a standardTracking implementation of the basic component.
Code lines: 20
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_basicstandardtracking_count
Description: Return a count of the serialization of a standardTracking implementation of the basic component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_basicstandardtracking_deserialize_rates
Description: Serialize rates of a standardTracking implementation of the basic component.
source code documentation

module galacticus_nodes

subroutine node_component_basicstandardtracking_deserialize_scales
Description: Serialize scales of a standardTracking implementation of the basic component.
Code lines: 14
Contained by: module galacticus_nodes

subroutine node_component_basicstandardtracking_deserialize_values
Description: Serialize values of a standardTracking implementation of the basic component.
Code lines: 14
Contained by: module galacticus_nodes

subroutine node_component_basicstandardtracking_destroy
Description: Destroy a standardTracking implementation of the basic component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine node_component_basicstandardtracking_dump
Description: Dump the contents of a standardTracking implementation of the basic component.
Code lines: 17
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine node_component_basicstandardtracking_dump_raw
Description: Dump the contents of a standardTracking implementation of the basic component in binary.
Code lines: 9
Contained by: module galacticus_nodes

subroutine node_component_basicstandardtracking_dump_xml
Description: Dump the contents of a standardTracking implementation of the basic component to XML.
Code lines: 11
Contained by: module galacticus_nodes

subroutine node_component_basicstandardtracking_initializor
Description: Initialize a standardTracking implementation of the basic component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

function node_component_basicstandardtracking_is_active
Description: Return true if the standardTracking implementation of the basic component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine node_component_basicstandardtracking_name_from_index
19.1. Program units

**Description:** Return the name of the property of given index for a standardTracking implementation of the basic component.

**Code lines:** 11

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**subroutine:** `node_component_basicstandardtracking_ode_step_rates_init`

**Description:** Initialize rates in a standardTracking implementation of the basic component for an ODE solver step.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_basicstandardtracking_odestepscalesinit`

**Description:** Initialize scales in a standardTracking implementation of the basic component for an ODE solver step.

**Code lines:** 6

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_basicstandardtracking_output_count`

**Description:** Increment output property count for a standardTracking implementation of the basic component.

**Code lines:** 10

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_basicstandardtracking_output_names`

**Description:** Establish property names for a standardTracking implementation of the basic component.

**Code lines:** 12

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_basicstandardtracking_read_raw`

**Description:** Read the contents of a standardTracking implementation of the basic component in binary.

**Code lines:** 10

**Contained by:** module `galacticus_nodes`

**Modules used:** `memory_management`

**subroutine:** `node_component_basicstandardtracking_serialize_rates`

**Description:** Serialize rates of a standardTracking implementation of the basic component.

**Code lines:** 14

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_basicstandardtracking_serialize_scales`

**Description:** Serialize scales of a standardTracking implementation of the basic component.

**Code lines:** 14

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_basicstandardtracking_serialize_values`

**Description:** Serialize values of a standardTracking implementation of the basic component.

**Code lines:** 14

**Contained by:** module `galacticus_nodes`
function: node_component_basicstandardtracking_type
Description: Returns the type for the standardTracking implementation of the basic component.
Code lines: 8
Contained by: module galacticus_nodes

function: node_component_black_hole_simple_enclosed_mass
Description: Computes the mass within a given radius for a central black hole. Black hole is treated as a point mass.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: galactic_structure_options

function: node_component_black_hole_simple_seed_mass
Description: Return the seed mass for simple black holes.
Code lines: 32
Contained by: module galacticus_nodes
Modules used: input_parameters

function: node_component_black_hole_standard_enclosed_mass
Description: Computes the mass within a given radius for a central black hole. Black hole is treated as a point mass.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: galactic_structure_options

function: node_component_black_hole_standard_seed_mass
Description: Return the seed mass for standard black holes.
Code lines: 32
Contained by: module galacticus_nodes
Modules used: input_parameters

function: node_component_black_hole_standard_seed_spin
Description: Return the seed spin for standard black holes.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: input_parameters

function: node_component_black_hole_standard_spin
Description: Return the spin of a standard black hole.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_blackhole_builder
Description: Build a generic blackHole component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error
19.1. Program units

**subroutine**: node_component_blackhole_destroy
  *Description*: Destroys a blackHole component.
  *Code lines*: 7
  *Contained by*: module galacticus_nodes

**subroutine**: node_component_blackhole_dump
  *Description*: Dump the contents of a generic blackHole component.
  *Code lines*: 10
  *Contained by*: module galacticus_nodes
  *Modules used*: galacticus_display iso_varying_string

**subroutine**: node_component_blackhole_initializor
  *Description*: Initialize a generic blackHole component.
  *Code lines*: 8
  *Contained by*: module galacticus_nodes
  *Modules used*: galacticus_error

**subroutine**: node_component_blackhole_move
  *Description*: Move instances of the blackHole component, from one node to another.
  *Code lines*: 81
  *Contained by*: module galacticus_nodes
  *Modules used*: galacticus_error

**subroutine**: node_component_blackhole_output
  *Description*: Output properties for a blackHole component.
  *Code lines*: 21
  *Contained by*: module galacticus_nodes

**subroutine**: node_component_blackhole_output_count
  *Description*: Increment the count of properties to output for a generic blackHole component.
  *Code lines*: 13
  *Contained by*: module galacticus_nodes

**subroutine**: node_component_blackhole_output_names
  *Description*: Establish property names for a generic blackHole component.
  *Code lines*: 15
  *Contained by*: module galacticus_nodes

**subroutine**: node_component_blackhole_remove
  *Description*: Removes an instance of the blackHole component, shifting other instances to keep the array contiguous.
  *Code lines*: 47
  *Contained by*: module galacticus_nodes
  *Modules used*: galacticus_error

**function**: node_component_blackhole_type
  *Description*: Returns the type for the blackHole component.
  *Code lines*: 8
19. Source Code Documentation

**module** galacticus_nodes

**subroutine**: node_component_blackholenull_builder

*Description:* Build a null implementation of the blackHole component.

*Code lines:* 10

*Contained by:* module galacticus_nodes

*Modules used:*

- fox_dom
- galacticus_error
- memory_management

**function**: node_component_blackholenull_count

*Description:* Return a count of the serialization of a null implementation of the blackHole component.

*Code lines:* 7

*Contained by:* module galacticus_nodes

**subroutine**: node_component_blackholenull_deserialize_rates

*Description:* Serialize rates of a null implementation of the blackHole component.

*Code lines:* 7

*Contained by:* module galacticus_nodes

**subroutine**: node_component_blackholenull_deserialize_scales

*Description:* Serialize scales of a null implementation of the blackHole component.

*Code lines:* 7

*Contained by:* module galacticus_nodes

**subroutine**: node_component_blackholenull_deserialize_values

*Description:* Serialize values of a null implementation of the blackHole component.

*Code lines:* 7

*Contained by:* module galacticus_nodes

**subroutine**: node_component_blackholenull_destroy

*Description:* Destroy a null implementation of the blackHole component.

*Code lines:* 7

*Contained by:* module galacticus_nodes

*Modules used:* memory_management

**subroutine**: node_component_blackholenull_dump

*Description:* Dump the contents of a null implementation of the blackHole component.

*Code lines:* 9

*Contained by:* module galacticus_nodes

*Modules used:*

- galacticus_display
- iso_varying_string
- string_handling

**subroutine**: node_component_blackholenull_dump_raw

*Description:* Dump the contents of a null implementation of the blackHole component in binary.

*Code lines:* 7

*Contained by:* module galacticus_nodes

**subroutine**: node_component_blackholenull_dump_xml

*Description:* Dump the contents of a null implementation of the blackHole component to XML.

1486
19.1. Program units

Code lines: 7
Contained by: module galacticus_nodes

**subroutine**: node_component_blackholenull_initializor
*Description*: Initialize a null implementation of the blackHole component.
*Code lines*: 7
*Contained by*: module galacticus_nodes
*Modules used*: memory_management

**function**: node_component_blackholenull_is_active
*Description*: Return true if the null implementation of the blackHole component is the active choice.
*Code lines*: 6
*Contained by*: module galacticus_nodes

**subroutine**: node_component_blackholenull_name_from_index
*Description*: Return the name of the property of given index for a null implementation of the blackHole component.
*Code lines*: 9
*Contained by*: module galacticus_nodes
*Modules used*: iso_varying_string

**subroutine**: node_component_blackholenull_ode_step_rates_init
*Description*: Initialize rates in a null implementation of the blackHole component for an ODE solver step.
*Code lines*: 6
*Contained by*: module galacticus_nodes

**subroutine**: node_component_blackholenull_odestepscalesinit
*Description*: Initialize scales in a null implementation of the blackHole component for an ODE solver step.
*Code lines*: 5
*Contained by*: module galacticus_nodes

**subroutine**: node_component_blackholenull_output_count
*Description*: Increment output property count for a null implementation of the blackHole component.
*Code lines*: 9
*Contained by*: module galacticus_nodes

**subroutine**: node_component_blackholenull_output_names
*Description*: Establish property names for a null implementation of the blackHole component.
*Code lines*: 11
*Contained by*: module galacticus_nodes

**subroutine**: node_component_blackholenull_read_raw
*Description*: Read the contents of a null implementation of the blackHole component in binary.
*Code lines*: 8
*Contained by*: module galacticus_nodes
*Modules used*: memory_management

**subroutine**: node_component_blackholenull_serialize_rates
19. Source Code Documentation

Description: Serialize rates of a null implementation of the blackHole component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_blackholenull_serialize_scales
Description: Serialize scales of a null implementation of the blackHole component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_blackholenull_serialize_values
Description: Serialize values of a null implementation of the blackHole component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_blackholenull_type
Description: Returns the type for the null implementation of the blackHole component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_blackholesimple_builder
Description: Build a simple implementation of the blackHole component.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: fox_dom memory_management galacticus_error

function: node_component_blackholesimple_count
Description: Return a count of the serialization of a simple implementation of the blackHole component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_blackholesimple_deserialize_rates
Description: Serialize rates of a simple implementation of the blackHole component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_blackholesimple_deserialize_scales
Description: Serialize scales of a simple implementation of the blackHole component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_blackholesimple_deserialize_values
Description: Serialize values of a simple implementation of the blackHole component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_blackholesimple_destroy
Description: Destroy a simple implementation of the blackHole component.
Code lines: 7
19.1. Program units

*Contained by:* module `galacticus_nodes`

*Modules used:* `memory_management`

**subroutine:** `node_component_blackholesimple_dump`

*Description:* Dump the contents of a simple implementation of the blackHole component.

*Code lines:* 16

*Contained by:* module `galacticus_nodes`

*Modules used:* `memory_management`

**subroutine:** `node_component_blackholesimple_dump_raw`

*Description:* Dump the contents of a simple implementation of the blackHole component in binary.

*Code lines:* 8

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_blackholesimple_dump_xml`

*Description:* Dump the contents of a simple implementation of the blackHole component to XML.

*Code lines:* 11

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_blackholesimple_initializor`

*Description:* Initialize a simple implementation of the blackHole component.

*Code lines:* 8

*Contained by:* module `galacticus_nodes`

*Modules used:* `memory_management`

**function:** `node_component_blackholesimple_is_active`

*Description:* Return true if the simple implementation of the blackHole component is the active choice.

*Code lines:* 6

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_blackholesimple_name_from_index`

*Description:* Return the name of the property of given index for a simple implementation of the blackHole component.

*Code lines:* 14

*Contained by:* module `galacticus_nodes`

*Modules used:* `iso_varying_string`

**subroutine:** `node_component_blackholesimple_ode_step_rates_init`

*Description:* Initialize rates in a simple implementation of the blackHole component for an ODE solver step.

*Code lines:* 7

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_blackholesimple_odestepscalesinit`

*Description:* Initialize scales in a simple implementation of the blackHole component for an ODE solver step.

*Code lines:* 6

*Contained by:* module `galacticus_nodes`
subroutine: node_component_blackholesimple_output_count
Description: Increment output property count for a simple implementation of the blackHole component.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_blackholesimple_output_names
Description: Establish property names for a simple implementation of the blackHole component.
Code lines: 24
Contained by: module galacticus_nodes

subroutine: node_component_blackholesimple_read_raw
Description: Read the contents of a simple implementation of the blackHole component in binary.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_blackholesimple_serialize_rates
Description: Serialize rates of a simple implementation of the blackHole component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_blackholesimple_serialize_scales
Description: Serialize scales of a simple implementation of the blackHole component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_blackholesimple_serialize_values
Description: Serialize values of a simple implementation of the blackHole component.
Code lines: 11
Contained by: module galacticus_nodes

function: node_component_blackholesimple_type
Description: Returns the type for the simple implementation of the blackHole component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_blackholestandard_builder
Description: Build a standard implementation of the blackHole component.
Code lines: 37
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_blackholestandard_count
Description: Return a count of the serialization of a standard implementation of the blackHole component.
Code lines: 8
Contained by: module galacticus_nodes
subroutine: node_component_blackholestandard_deserialize_rates
Description: Serialize rates of a standard implementation of the blackHole component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_blackholestandard_deserialize_scales
Description: Serialize scales of a standard implementation of the blackHole component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_blackholestandard_deserialize_values
Description: Serialize values of a standard implementation of the blackHole component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_blackholestandard_destroy
Description: Destroy a standard implementation of the blackHole component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_blackholestandard_dump
Description: Dump the contents of a standard implementation of the blackHole component.
Code lines: 25
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_blackholestandard_dump_raw
Description: Dump the contents of a standard implementation of the blackHole component in binary.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_blackholestandard_dump_xml
Description: Dump the contents of a standard implementation of the blackHole component to XML.
Code lines: 17
Contained by: module galacticus_nodes

subroutine: node_component_blackholestandard_initializor
Description: Initialize a standard implementation of the blackHole component.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_blackholestandard_is_active
Description: Return true if the standard implementation of the blackHole component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes
subroutine: node_component_blackholestandard_name_from_index
Description: Return the name of the property of given index for a standard implementation of the blackHole component.
Code lines: 24
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_blackholestandard_ode_step_rates_init
Description: Initialize rates in a standard implementation of the blackHole component for an ODE solver step.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_blackholestandard_odestepscalesinit
Description: Initialize scales in a standard implementation of the blackHole component for an ODE solver step.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_blackholestandard_output_count
Description: Increment output property count for a standard implementation of the blackHole component.
Code lines: 12
Contained by: module galacticus_nodes

subroutine: node_component_blackholestandard_output_names
Description: Establish property names for a standard implementation of the blackHole component.
Code lines: 39
Contained by: module galacticus_nodes

subroutine: node_component_blackholestandard_read_raw
Description: Read the contents of a standard implementation of the blackHole component in binary.
Code lines: 12
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_blackholestandard_serialize_rates
Description: Serialize rates of a standard implementation of the blackHole component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_blackholestandard_serialize_scales
Description: Serialize scales of a standard implementation of the blackHole component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_blackholestandard_serialize_values
Description: Serialize values of a standard implementation of the blackHole component.
Code lines: 15
19.1. Program units

function: node_component_blackholestandard_type
Description: Returns the type for the standard implementation of the blackHole component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofile_builder
Description: Build a generic darkMatterProfile component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error

subroutine: node_component_darkmatterprofile_destroy
Description: Destroys a darkMatterProfile component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofile_dump
Description: Dump the contents of a generic darkMatterProfile component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string

subroutine: node_component_darkmatterprofile_initializor
Description: Initialize a generic darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_darkmatterprofile_move
Description: Move instances of the darkMatterProfile component, from one node to another.
Code lines: 95
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_darkmatterprofile_output
Description: Output properties for a darkMatterProfile component.
Code lines: 23
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofile_output_count
Description: Increment the count of properties to output for a generic darkMatterProfile component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofile_output_names
Description: Establish property names for a generic darkMatterProfile component.
Code lines: 15
19. Source Code Documentation

**module** galacticus_nodes

**subroutine**: node_component_darkmatterprofile_remove

**Description**: Removes an instance of the darkMatterProfile component, shifting other instances to keep the array contiguous.

**Code lines**: 55

**Contained by**: module galacticus_nodes

**Modules used**: galacticus_error

**function**: node_component_darkmatterprofile_type

**Description**: Returns the type for the darkMatterProfile component.

**Code lines**: 8

**Contained by**: module galacticus_nodes

**subroutine**: node_component_darkmatterprofilenull_builder

**Description**: Build a null implementation of the darkMatterProfile component.

**Code lines**: 10

**Contained by**: module galacticus_nodes

**Modules used**: fox_dom galacticus_error memory_management

**function**: node_component_darkmatterprofilenull_count

**Description**: Return a count of the serialization of a null implementation of the darkMatterProfile component.

**Code lines**: 7

**Contained by**: module galacticus_nodes

**subroutine**: node_component_darkmatterprofilenull_deserialize_rates

**Description**: Serialize rates of a null implementation of the darkMatterProfile component.

**Code lines**: 7

**Contained by**: module galacticus_nodes

**subroutine**: node_component_darkmatterprofilenull_deserialize_scales

**Description**: Serialize scales of a null implementation of the darkMatterProfile component.

**Code lines**: 7

**Contained by**: module galacticus_nodes

**subroutine**: node_component_darkmatterprofilenull_deserialize_values

**Description**: Serialize values of a null implementation of the darkMatterProfile component.

**Code lines**: 7

**Contained by**: module galacticus_nodes

**subroutine**: node_component_darkmatterprofilenull_destroy

**Description**: Destroy a null implementation of the darkMatterProfile component.

**Code lines**: 7

**Contained by**: module galacticus_nodes

**Modules used**: memory_management

**subroutine**: node_component_darkmatterprofilenull_dump
19.1. Program units

Description: Dump the contents of a null implementation of the darkMatterProfile component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_darkmatterprofilenull_dump_raw
Description: Dump the contents of a null implementation of the darkMatterProfile component in binary.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilenull_dump_xml
Description: Dump the contents of a null implementation of the darkMatterProfile component to XML.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilenull_initializor
Description: Initialize a null implementation of the darkMatterProfile component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_darkmatterprofilenull_is_active
Description: Return true if the null implementation of the darkMatterProfile component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilenull_name_from_index
Description: Return the name of the property of given index for a null implementation of the darkMatterProfile component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_darkmatterprofilenull_ode_step_rates_init
Description: Initialize rates in a null implementation of the darkMatterProfile component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilenull_odestepscalesinit
Description: Initialize scales in a null implementation of the darkMatterProfile component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilenull_output_count
Description: Increment output property count for a null implementation of the darkMatterProfile component.
19. Source Code Documentation

subroutine: node_component_darkmatterprofilenull_output_names
Description: Establish property names for a null implementation of the darkMatterProfile component.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilenull_read_raw
Description: Read the contents of a null implementation of the darkMatterProfile component in binary.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_darkmatterprofilenull_serialize_rates
Description: Serialize rates of a null implementation of the darkMatterProfile component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilenull_serialize_scales
Description: Serialize scales of a null implementation of the darkMatterProfile component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilenull_serialize_values
Description: Serialize values of a null implementation of the darkMatterProfile component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_darkmatterprofilenull_type
Description: Returns the type for the null implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescale_builder
Description: Build a scale implementation of the darkMatterProfile component.
Code lines: 25
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_darkmatterprofilescale_count
Description: Return a count of the serialization of a scale implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescale_deserialize_rates
Description: Serialize rates of a scale implementation of the darkMatterProfile component.
19.1. Program units

Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescale_deserialize_scales
Description: Serialize scales of a scale implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescale_deserialize_values
Description: Serialize values of a scale implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescale_destroy
Description: Destroy a scale implementation of the darkMatterProfile component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_darkmatterprofilescale_dump
Description: Dump the contents of a scale implementation of the darkMatterProfile component.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_darkmatterprofilescale_dump_raw
Description: Dump the contents of a scale implementation of the darkMatterProfile component in binary.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescale_dump_xml
Description: Dump the contents of a scale implementation of the darkMatterProfile component to XML.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescale_initializor
Description: Initialize a scale implementation of the darkMatterProfile component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_darkmatterprofilescale_is_active
Description: Return true if the scale implementation of the darkMatterProfile component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescale_name_from_index
19. Source Code Documentation

**Description:** Return the name of the property of given index for a scale implementation of the darkMatterProfile component.

**Code lines:** 14

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**subroutine:** `node_component_darkmatterprofilescale_ode_step_rates_init`

**Description:** Initialize rates in a scale implementation of the darkMatterProfile component for an ODE solver step.

**Code lines:** 7

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_darkmatterprofilescale_odestepscalesinit`

**Description:** Initialize scales in a scale implementation of the darkMatterProfile component for an ODE solver step.

**Code lines:** 6

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_darkmatterprofilescale_output_count`

**Description:** Increment output property count for a scale implementation of the darkMatterProfile component.

**Code lines:** 10

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_darkmatterprofilescale_output_names`

**Description:** Establish property names for a scale implementation of the darkMatterProfile component.

**Code lines:** 24

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_darkmatterprofilescale_read_raw`

**Description:** Read the contents of a scale implementation of the darkMatterProfile component in binary.

**Code lines:** 10

**Contained by:** module `galacticus_nodes`

**Modules used:** `memory_management`

**subroutine:** `node_component_darkmatterprofilescale_serialize_rates`

**Description:** Serialize rates of a scale implementation of the darkMatterProfile component.

**Code lines:** 11

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_darkmatterprofilescale_serialize_scales`

**Description:** Serialize scales of a scale implementation of the darkMatterProfile component.

**Code lines:** 11

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_darkmatterprofilescale_serialize_values`

**Description:** Serialize values of a scale implementation of the darkMatterProfile component.

**Code lines:** 11

**Contained by:** module `galacticus_nodes`
function: node_component_darkmatterprofilescalescale_type
Description: Returns the type for the scale implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescalescalepreset_builder
Description: Build a scalePreset implementation of the darkMatterProfile component.
Code lines: 25
Contained by: module galacticus_nodes
Modules used: fox_dom memory_management galacticus_error

function: node_component_darkmatterprofilescalescalepreset_count
Description: Return a count of the serialization of a scalePreset implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescalescalepreset_deserialize_rates
Description: Serialize rates of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescalescalepreset_deserialize_scales
Description: Serialize scales of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescalescalepreset_deserialize_values
Description: Serialize values of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescalescalepreset_destroy
Description: Destroy a scalePreset implementation of the darkMatterProfile component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_darkmatterprofilescalescalepreset_dump
Description: Dump the contents of a scalePreset implementation of the darkMatterProfile component.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_darkmatterprofilescalescalepreset_dump_raw
Description: Dump the contents of a scalePreset implementation of the darkMatterProfile component in binary.
19. Source Code Documentation

subroutine: node_component_darkmatterprofilescalepreset_dump_xml
Description: Dump the contents of a scalePreset implementation of the darkMatterProfile component to XML.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescalepreset_initializor
Description: Initialize a scalePreset implementation of the darkMatterProfile component.
Code lines: 9
Contained by: module galacticus_nodes

function: node_component_darkmatterprofilescalepreset_is_active
Description: Return true if the scalePreset implementation of the darkMatterProfile component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescalepreset_name_from_index
Description: Return the name of the property of given index for a scalePreset implementation of the darkMatterProfile component.
Code lines: 14
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_darkmatterprofilescalepreset_ode_step_rates_init
Description: Initialize rates in a scalePreset implementation of the darkMatterProfile component for an ODE solver step.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescalepreset_odestepscalesinit
Description: Initialize scales in a scalePreset implementation of the darkMatterProfile component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescalepreset_output_count
Description: Increment output property count for a scalePreset implementation of the darkMatterProfile component.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescalepreset_output_names
Description: Establish property names for a scalePreset implementation of the darkMatterProfile component.
Code lines: 24
Contained by: module galacticus_nodes
subroutine: node_component_darkmatterprofilescaleshape_builder
Description: Build a scaleShape implementation of the darkMatterProfile component.
Code lines: 26
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_darkmatterprofilescaleshape_count
Description: Return a count of the serialization of a scaleShape implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_deserialize_rates
Description: Serialize rates of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_deserialize_scales
Description: Serialize scales of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

function: node_component_darkmatterprofilescaleshape_deserialize_values
Description: Serialize values of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_serialize_rates
Description: Serialize rates of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_serialize_scales
Description: Serialize scales of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_serialize_values
Description: Serialize values of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

function: node_component_darkmatterprofilescaleshape_serialize_type
Description: Returns the type for the scalePreset implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_builder
Description: Build a scaleShape implementation of the darkMatterProfile component.
Code lines: 26
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_darkmatterprofilescaleshape_count
Description: Return a count of the serialization of a scaleShape implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_deserialize_rates
Description: Serialize rates of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_deserialize_scales
Description: Serialize scales of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_deserialize_values
Description: Serialize values of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

function: node_component_darkmatterprofilescaleshape_deserialize_type
Description: Returns the type for the scaleShape implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_serialize_rates
Description: Serialize rates of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_serialize_scales
Description: Serialize scales of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_serialize_values
Description: Serialize values of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

function: node_component_darkmatterprofilescaleshape_serialize_type
Description: Returns the type for the scaleShape implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_builder
Description: Build a scaleShape implementation of the darkMatterProfile component.
Code lines: 26
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_darkmatterprofilescaleshape_count
Description: Return a count of the serialization of a scaleShape implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_deserialize_rates
Description: Serialize rates of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_deserialize_scales
Description: Serialize scales of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_deserialize_values
Description: Serialize values of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

function: node_component_darkmatterprofilescaleshape_deserialize_type
Description: Returns the type for the scaleShape implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_serialize_rates
Description: Serialize rates of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_serialize_scales
Description: Serialize scales of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_serialize_values
Description: Serialize values of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

function: node_component_darkmatterprofilescaleshape_serialize_type
Description: Returns the type for the scaleShape implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_builder
Description: Build a scaleShape implementation of the darkMatterProfile component.
Code lines: 26
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_darkmatterprofilescaleshape_count
Description: Return a count of the serialization of a scaleShape implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_deserialize_rates
Description: Serialize rates of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_deserialize_scales
Description: Serialize scales of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_deserialize_values
Description: Serialize values of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

function: node_component_darkmatterprofilescaleshape_deserialize_type
Description: Returns the type for the scaleShape implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_serialize_rates
Description: Serialize rates of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_serialize_scales
Description: Serialize scales of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_serialize_values
Description: Serialize values of a scalePreset implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

function: node_component_darkmatterprofilescaleshape_serialize_type
Description: Returns the type for the scaleShape implementation of the darkMatterProfile component.
Code lines: 8
Contained by: module galacticus_nodes
19. Source Code Documentation

**Contained by:** module galacticus_nodes

**subroutine:** node_component_darkmatterprofilescaleshape_deserialize_values
**Description:** Serialize values of a scaleShape implementation of the darkMatterProfile component.
**Code lines:** 16
**Contained by:** module galacticus_nodes

**subroutine:** node_component_darkmatterprofilescaleshape_destroy
**Description:** Destroy a scaleShape implementation of the darkMatterProfile component.
**Code lines:** 7
**Contained by:** module galacticus_nodes
**Modules used:** memory_management

**subroutine:** node_component_darkmatterprofilescaleshape_dump
**Description:** Dump the contents of a scaleShape implementation of the darkMatterProfile component.
**Code lines:** 20
**Contained by:** module galacticus_nodes
**Modules used:** galacticus_display iso_varying_string string_handling

**subroutine:** node_component_darkmatterprofilescaleshape_dump_raw
**Description:** Dump the contents of a scaleShape implementation of the darkMatterProfile component in binary.
**Code lines:** 10
**Contained by:** module galacticus_nodes

**subroutine:** node_component_darkmatterprofilescaleshape_dump_xml
**Description:** Dump the contents of a scaleShape implementation of the darkMatterProfile component to XML.
**Code lines:** 12
**Contained by:** module galacticus_nodes

**subroutine:** node_component_darkmatterprofilescaleshape_initializor
**Description:** Initialize a scaleShape implementation of the darkMatterProfile component.
**Code lines:** 10
**Contained by:** module galacticus_nodes
**Modules used:** memory_management

**function:** node_component_darkmatterprofilescaleshape_is_active
**Description:** Return true if the scaleShape implementation of the darkMatterProfile component is the active choice.
**Code lines:** 6
**Contained by:** module galacticus_nodes

**subroutine:** node_component_darkmatterprofilescaleshape_name_from_index
**Description:** Return the name of the property of given index for a scaleShape implementation of the darkMatterProfile component.
**Code lines:** 16
**Contained by:** module galacticus_nodes
**Modules used:** iso_varying_string
subroutine: node_component_darkmatterprofilescaleshape_ode_step_rates_init
Description: Initialize rates in a scaleShape implementation of the darkMatterProfile component for an ODE solver step.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_odestepscalesinit
Description: Initialize scales in a scaleShape implementation of the darkMatterProfile component for an ODE solver step.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_output_count
Description: Increment output property count for a scaleShape implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_output_names
Description: Establish property names for a scaleShape implementation of the darkMatterProfile component.
Code lines: 25
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_read_raw
Description: Read the contents of a scaleShape implementation of the darkMatterProfile component in binary.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_darkmatterprofilescaleshape_serialize_rates
Description: Serialize rates of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_serialize_scales
Description: Serialize scales of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilescaleshape_serialize_values
Description: Serialize values of a scaleShape implementation of the darkMatterProfile component.
Code lines: 16
Contained by: module galacticus_nodes

function: node_component_darkmatterprofilescaleshape_type
Description: Returns the type for the scaleShape implementation of the darkMatterProfile component.
19. Source Code Documentation

Code lines: 8
Contained by: module galacticus_nodes

**function: node_component_density_null**
Description: A null implementation of the density in a component. Always returns zero.
Code lines: 10
Contained by: module galacticus_nodes

**subroutine: node_component_deserialize_null**
Description: Deserialize a generic treenode component.
Code lines: 7
Contained by: module galacticus_nodes

**subroutine: node_component_disk_builder**
Description: Build a generic disk component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error

**subroutine: node_component_disk_destroy**
Description: Destroys a disk component.
Code lines: 7
Contained by: module galacticus_nodes

**subroutine: node_component_disk_dump**
Description: Dump the contents of a generic disk component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string

**subroutine: node_component_disk_exponential_attach_pipes**
Description: Attach cooling pipes to the exponential disk component.
Code lines: 14
Contained by: module galacticus_nodes
Modules used: galacticus_error

**function: node_component_disk_exponential_density**
Description: Computes the density at a given position for an exponential disk.
Code lines: 52
Contained by: module galacticus_nodes
Modules used: coordinate_systems galactic_structure_options	node_component_disk_exponential_data numerical_constants_math
tables

**function: node_component_disk_exponential_enclosed_mass**
Description: Computes the mass within a given radius for an exponential disk.
Code lines: 45
Contained by: module galacticus_nodes
Modules used: galactic_structure_options node_component_disk_exponential_data
19.1. Program units

tables

**function**: node_component_disk_exponential_half_mass_radius
*Description*: Return the half-mass radius of the exponential disk.
*Code lines*: 8
*Contained by*: module galacticus_nodes

**function**: node_component_disk_exponential_potential
*Description*: Compute the gravitational potential due to an exponential disk.
*Code lines*: 42
*Contained by*: module galacticus_nodes
*Modules used*: bessel_functions, galactic_structure_options, node_component_disk_exponential_data, numerical_constants_physical, tables

**function**: node_component_disk_exponential_rotation_curve
*Description*: Computes the rotation curve at a given radius for an exponential disk.
*Code lines*: 45
*Contained by*: module galacticus_nodes
*Modules used*: galactic_structure_options, node_component_disk_exponential_data, numerical_constants_physical, tables

**function**: node_component_disk_exponential_rotation_curve_gradient
*Description*: Computes the rotation curve gradient for an exponential disk.
*Code lines*: 33
*Contained by*: module galacticus_nodes
*Modules used*: galactic_structure_options, node_component_disk_exponential_data, numerical_constants_physical, tables

dependencies: galacticus_error

**function**: node_component_disk_exponential_surface_density
*Description*: Computes the surface density at a given position for an exponential disk.
*Code lines*: 67
*Contained by*: module galacticus_nodes
*Modules used*: galactic_structure_options, node_component_disk_exponential_data, numerical_constants_math, tables

**subroutine**: node_component_disk_initializor
*Description*: Initialize a generic disk component.
*Code lines*: 8
*Contained by*: module galacticus_nodes
*Modules used*: galacticus_error

**subroutine**: node_component_disk_move
*Description*: Move instances of the disk component, from one node to another.
*Code lines*: 81
*Contained by*: module galacticus_nodes
*Modules used*: galacticus_error
subroutine: node_component_disk_output
Description: Output properties for a disk component.
Code lines: 41
Contained by: module galacticus_nodes

subroutine: node_component_disk_output_count
Description: Increment the count of properties to output for a generic disk component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_disk_output_names
Description: Establish property names for a generic disk component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_disk_remove
Description: Removes an instance of the disk component, shifting other instances to keep the array contiguous.
Code lines: 47
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: node_component_disk_type
Description: Returns the type for the disk component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_disk_very_simple_attach_pipe
Description: Attach cooling pipes to the very simple disk component.
Code lines: 12
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: node_component_disk_very_simple_enclosed_mass
Description: Computes the mass within a given radius for an very simple disk.
Code lines: 34
Contained by: module galacticus_nodes
Modules used: galacticus_error galactic_structure_options

subroutine: node_component_disk_exponential_builder
Description: Build a exponential implementation of the disk component.
Code lines: 79
Contained by: module galacticus_nodes
Modules used: fox_dom memory_management galacticus_error

function: node_component_disk_exponential_count
Description: Return a count of the serialization of a exponential implementation of the disk component.
19.1. Program units

Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_diskexponential_deserialize_rates
Description: Serialize rates of a exponential implementation of the disk component.
Code lines: 30
Contained by: module galacticus_nodes

subroutine: node_component_diskexponential_deserialize_scales
Description: Serialize scales of a exponential implementation of the disk component.
Code lines: 30
Contained by: module galacticus_nodes

subroutine: node_component_diskexponential_deserialize_values
Description: Serialize values of a exponential implementation of the disk component.
Code lines: 30
Contained by: module galacticus_nodes

subroutine: node_component_diskexponential_destroy
Description: Destroy a exponential implementation of the disk component.
Code lines: 22
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_diskexponential_dump
Description: Dump the contents of a exponential implementation of the disk component.
Code lines: 51
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_diskexponential_dump_raw
Description: Dump the contents of a exponential implementation of the disk component in binary.
Code lines: 18
Contained by: module galacticus_nodes

subroutine: node_component_diskexponential_dump_xml
Description: Dump the contents of a exponential implementation of the disk component to XML.
Code lines: 27
Contained by: module galacticus_nodes

subroutine: node_component_diskexponential_initializor
Description: Initialize a exponential implementation of the disk component.
Code lines: 18
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_diskexponential_is_active
Description: Return true if the exponential implementation of the disk component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_diskexponential_name_from_index
Description: Return the name of the property of given index for a exponential implementation of the disk component.
Code lines: 49
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_diskexponential_ode_step_rates_init
Description: Initialize rates in a exponential implementation of the disk component for an ODE solver step.
Code lines: 14
Contained by: module galacticus_nodes

subroutine: node_component_diskexponential_ode_stepscalesinit
Description: Initialize scales in a exponential implementation of the disk component for an ODE solver step.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_diskexponential_output_count
Description: Increment output property count for a exponential implementation of the disk component.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: node_component_diskexponential_output_names
Description: Establish property names for a exponential implementation of the disk component.
Code lines: 99
Contained by: module galacticus_nodes

subroutine: node_component_diskexponential_read_raw
Description: Read the contents of a exponential implementation of the disk component in binary.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_diskexponential_serialize_rates
Description: Serialize rates of a exponential implementation of the disk component.
Code lines: 30
Contained by: module galacticus_nodes

subroutine: node_component_diskexponential_serialize_scales
Description: Serialize scales of a exponential implementation of the disk component.
Code lines: 30
Contained by: module galacticus_nodes

subroutine: node_component_diskexponential_serialize_values
19.1. Program units

**Description:** Serialize values of a exponential implementation of the disk component.

**Code lines:** 30

**Contained by:** module galacticus_nodes

**function:** node_component_diskexponential_type

**Description:** Returns the type for the exponential implementation of the disk component.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**subroutine:** node_component_disknull_builder

**Description:** Build a null implementation of the disk component.

**Code lines:** 10

**Contained by:** module galacticus_nodes

**Modules used:** fox_dom galacticus_error

memory_management

**function:** node_component_disknull_count

**Description:** Return a count of the serialization of a null implementation of the disk component.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**subroutine:** node_component_disknull_deserialize_rates

**Description:** Serialize rates of a null implementation of the disk component.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**subroutine:** node_component_disknull_deserialize_scales

**Description:** Serialize scales of a null implementation of the disk component.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**subroutine:** node_component_disknull_deserialize_values

**Description:** Serialize values of a null implementation of the disk component.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**subroutine:** node_component_disknull_destroy

**Description:** Destroy a null implementation of the disk component.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**Modules used:** memory_management

**subroutine:** node_component_disknull_dump

**Description:** Dump the contents of a null implementation of the disk component.

**Code lines:** 9

**Contained by:** module galacticus_nodes

**Modules used:** galacticus_display iso_varying_string

string_handling
subroutine: node_component_disknull_dump_raw
Description: Dump the contents of a null implementation of the disk component in binary.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_disknull_dump_xml
Description: Dump the contents of a null implementation of the disk component to XML.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_disknull_initializor
Description: Initialize a null implementation of the disk component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_disknull_is_active
Description: Return true if the null implementation of the disk component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_disknull_name_from_index
Description: Return the name of the property of given index for a null implementation of the disk component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_disknull_ode_step_rates_init
Description: Initialize rates in a null implementation of the disk component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_disknull_odestepscalesinit
Description: Initialize scales in a null implementation of the disk component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_disknull_output_count
Description: Increment output property count for a null implementation of the disk component.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_disknull_output_names
Description: Establish property names for a null implementation of the disk component.
Code lines: 11
Contained by: module galacticus_nodes
subroutine: node_component_disknull_read_raw
Description: Read the contents of a null implementation of the disk component in binary.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_disknull_serialize_rates
Description: Serialize rates of a null implementation of the disk component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_disknull_serialize_scales
Description: Serialize scales of a null implementation of the disk component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_disknull_serialize_values
Description: Serialize values of a null implementation of the disk component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_disknull_type
Description: Returns the type for the null implementation of the disk component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_diskverysimple_builder
Description: Build a verySimple implementation of the disk component.
Code lines: 25
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_diskverysimple_count
Description: Return a count of the serialization of a verySimple implementation of the disk component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_diskverysimple_deserialize_rates
Description: Serialize rates of a verySimple implementation of the disk component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_diskverysimple_deserialize_scales
Description: Serialize scales of a verySimple implementation of the disk component.
Code lines: 13
Contained by: module galacticus_nodes
19. Source Code Documentation

subroutine: node_component_diskverysimple_deserialize_values
  Description: Serialize values of a verySimple implementation of the disk component.
  Code lines: 13
  Contained by: module galacticus_nodes

subroutine: node_component_diskverysimple_destroy
  Description: Destroy a verySimple implementation of the disk component.
  Code lines: 7
  Contained by: module galacticus_nodes
  Modules used: memory_management

subroutine: node_component_diskverysimple_dump
  Description: Dump the contents of a verySimple implementation of the disk component.
  Code lines: 19
  Contained by: module galacticus_nodes
  Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_diskverysimple_dump_raw
  Description: Dump the contents of a verySimple implementation of the disk component in binary.
  Code lines: 9
  Contained by: module galacticus_nodes

subroutine: node_component_diskverysimple_dump_xml
  Description: Dump the contents of a verySimple implementation of the disk component to XML.
  Code lines: 11
  Contained by: module galacticus_nodes

subroutine: node_component_diskverysimple_initialize
  Description: Initialize a verySimple implementation of the disk component.
  Code lines: 9
  Contained by: module galacticus_nodes
  Modules used: memory_management

function: node_component_diskverysimple_is_active
  Description: Return true if the verySimple implementation of the disk component is the active choice.
  Code lines: 6
  Contained by: module galacticus_nodes

subroutine: node_component_diskverysimple_name_from_index
  Description: Return the name of the property of given index for a verySimple implementation of the disk component.
  Code lines: 19
  Contained by: module galacticus_nodes
  Modules used: iso_varying_string

subroutine: node_component_diskverysimple_ode_step_rates_init
  Description: Initialize rates in a verySimple implementation of the disk component for an ODE solver step.
19.1. Program units

Subroutine: node_component_diskverysimple_odestepscalesinit
Description: Initialize scales in a verySimple implementation of the disk component for an ODE solver step.
Code lines: 7
Contained by: module galacticus_nodes

Subroutine: node_component_diskverysimple_output_count
Description: Increment output property count for a verySimple implementation of the disk component.
Code lines: 10
Contained by: module galacticus_nodes

Subroutine: node_component_diskverysimple_output_names
Description: Establish property names for a verySimple implementation of the disk component.
Code lines: 37
Contained by: module galacticus_nodes

Subroutine: node_component_diskverysimple_read_raw
Description: Read the contents of a verySimple implementation of the disk component in binary.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: memory_management

Subroutine: node_component_diskverysimple_serialize_rates
Description: Serialize rates of a verySimple implementation of the disk component.
Code lines: 13
Contained by: module galacticus_nodes

Subroutine: node_component_diskverysimple_serialize_scales
Description: Serialize scales of a verySimple implementation of the disk component.
Code lines: 13
Contained by: module galacticus_nodes

Subroutine: node_component_diskverysimple_serialize_values
Description: Serialize values of a verySimple implementation of the disk component.
Code lines: 13
Contained by: module galacticus_nodes

Function: node_component_diskverysimple_type
Description: Returns the type for the verySimple implementation of the disk component.
Code lines: 8
Contained by: module galacticus_nodes

Subroutine: node_component_dump_null
Description: Dump a generic tree node component.
Code lines: 6
Contained by: module galacticus_nodes
19. Source Code Documentation

subroutine: node_component_dump_raw_null
Description: Dump a generic tree node component in binary.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_dump_xml_null
Description: Dump a generic tree node component to XML.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_dynamics_statistics_bars_record
Description: Record the dynamical state.
Code lines: 43
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_dynamicsstatistics_builder
Description: Build a generic dynamicsStatistics component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error

subroutine: node_component_dynamicsstatistics_destroy
Description: Destroys a dynamicsStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatistics_dump
Description: Dump the contents of a generic dynamicsStatistics component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string

subroutine: node_component_dynamicsstatistics_initializor
Description: Initialize a generic dynamicsStatistics component.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_dynamicsstatistics_move
Description: Move instances of the dynamicsStatistics component, from one node to another.
Code lines: 67
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_dynamicsstatistics_output
Description: Output properties for a dynamicsStatistics component.
Code lines: 11
19.1. Program units

Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatistics_output_count
  Description: Increment the count of properties to output for a generic dynamicsStatistics component.
  Code lines: 13
  Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatistics_output_names
  Description: Establish property names for a generic dynamicsStatistics component.
  Code lines: 15
  Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatistics_remove
  Description: Removes an instance of the dynamicsStatistics component, shifting other instances to keep
                the array contiguous.
  Code lines: 39
  Contained by: module galacticus_nodes
  Modules used: galacticus_error

function: node_component_dynamicsstatistics_type
  Description: Returns the type for the dynamicsStatistics component.
  Code lines: 8
  Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsbars_builder
  Description: Build a bars implementation of the dynamicsStatistics component.
  Code lines: 38
  Contained by: module galacticus_nodes
  Modules used: fox_dom galacticus_error memory_management

function: node_component_dynamicsstatisticsbars_count
  Description: Return a count of the serialization of a bars implementation of the dynamicsStatistics com-
                ponent.
  Code lines: 7
  Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsbars_deserialize_rates
  Description: Serialize rates of a bars implementation of the dynamicsStatistics component.
  Code lines: 7
  Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsbars_deserialize_scales
  Description: Serialize scales of a bars implementation of the dynamicsStatistics component.
  Code lines: 7
  Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsbars_deserialize_values
  Description: Serialize values of a bars implementation of the dynamicsStatistics component.
19. Source Code Documentation

subroutine: node_component_dynamicsstatisticsbars_destroy
Description: Destroy a bars implementation of the dynamicsStatistics component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_dynamicsstatisticsbars_dump
Description: Dump the contents of a bars implementation of the dynamicsStatistics component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_dynamicsstatisticsbars_dump_raw
Description: Dump the contents of a bars implementation of the dynamicsStatistics component to XML.
Code lines: 35
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_dynamicsstatisticsbars_dump_xml
Description: Dump the contents of a bars implementation of the dynamicsStatistics component in binary.
Code lines: 22
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsbars_initializor
Description: Initialize a bars implementation of the dynamicsStatistics component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_dynamicsstatisticsbars_is_active
Description: Return true if the bars implementation of the dynamicsStatistics component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsbars_name_from_index
Description: Return the name of the property of given index for a bars implementation of the dynamicsStatistics component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_dynamicsstatisticsbars_ode_step_rates_init
Description: Initialize rates in a bars implementation of the dynamicsStatistics component for an ODE solver step.
Code lines: 19
Contained by: module galacticus_nodes
subroutine: node_component_dynamicsstatisticsbars_odestep_scalesinit
Description: Initialize scales in a bars implementation of the dynamicsStatistics component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsbars_output_count
Description: Increment output property count for a bars implementation of the dynamicsStatistics component.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsbars_output_names
Description: Establish property names for a bars implementation of the dynamicsStatistics component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsbars_read_raw
Description: Read the contents of a bars implementation of the dynamicsStatistics component in binary.
Code lines: 28
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_dynamicsstatisticsbars_serialize_rates
Description: Serialize rates of a bars implementation of the dynamicsStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsbars_serialize_scales
Description: Serialize scales of a bars implementation of the dynamicsStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsbars_serialize_values
Description: Serialize values of a bars implementation of the dynamicsStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_dynamicsstatisticsbars_type
Description: Returns the type for the bars implementation of the dynamicsStatistics component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsnull_builder
Description: Build a null implementation of the dynamicsStatistics component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error
memory_management

function: node_component_dynamicsstatisticsnull_count
Description: Return a count of the serialization of a null implementation of the dynamicsStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsnull_deserialize_rates
Description: Serialize rates of a null implementation of the dynamicsStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsnull_deserialize_scales
Description: Serialize scales of a null implementation of the dynamicsStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsnull_deserialize_values
Description: Serialize values of a null implementation of the dynamicsStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsnull_destroy
Description: Destroy a null implementation of the dynamicsStatistics component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_dynamicsstatisticsnull_dump
Description: Dump the contents of a null implementation of the dynamicsStatistics component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_dynamicsstatisticsnull_dump_raw
Description: Dump the contents of a null implementation of the dynamicsStatistics component in binary.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsnull_dump_xml
Description: Dump the contents of a null implementation of the dynamicsStatistics component to XML.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsnull_initializor
Description: Initialize a null implementation of the dynamicsStatistics component.
Code lines: 7
19.1. Program units

```
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_dynamicsstatisticsnull_is_active
Description: Return true if the null implementation of the dynamicsStatistics component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsnull_name_from_index
Description: Return the name of the property of given index for a null implementation of the dynamicsStatistics component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_dynamicsstatisticsnull_ode_step_rates_init
Description: Initialize rates in a null implementation of the dynamicsStatistics component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsnull_odestepscalesinit
Description: Initialize scales in a null implementation of the dynamicsStatistics component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsnull_output_count
Description: Increment output property count for a null implementation of the dynamicsStatistics component.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsnull_output_names
Description: Establish property names for a null implementation of the dynamicsStatistics component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsnull_read_raw
Description: Read the contents of a null implementation of the dynamicsStatistics component in binary.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_dynamicsstatisticsnull_serialize_rates
Description: Serialize rates of a null implementation of the dynamicsStatistics component.
Code lines: 7
Contained by: module galacticus_nodes
```
subroutine: node_component_dynamicsstatisticsnull_serialize_scales
Description: Serialize scales of a null implementation of the dynamicsStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_dynamicsstatisticsnull_serialize_values
Description: Serialize values of a null implementation of the dynamicsStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_dynamicsstatisticsnull_type
Description: Returns the type for the null implementation of the dynamicsStatistics component.
Code lines: 8
Contained by: module galacticus_nodes

function: node_component_enclosed_mass_null
Description: A null implementation of the enclosed mass in a component. Always returns zero.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_formationtime_builder
Description: Build a generic formationTime component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error

subroutine: node_component_formationtime_destroy
Description: Destroys a formationTime component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_formationtime_dump
Description: Dump the contents of a generic formationTime component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string

subroutine: node_component_formationtime_initializor
Description: Initialize a generic formationTime component.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_formationtime_move
Description: Move instances of the formationTime component, from one node to another.
Code lines: 67
Contained by: module galacticus_nodes
19.1. Program units

Modules used: galacticus_error

**subroutine:** node_component_formationtime_output  
*Description:* Output properties for a formationTime component.  
*Code lines:* 11  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_formationtime_output_count  
*Description:* Increment the count of properties to output for a generic formationTime component.  
*Code lines:* 13  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_formationtime_output_names  
*Description:* Establish property names for a generic formationTime component.  
*Code lines:* 15  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_formationtime_remove  
*Description:* Removes an instance of the formationTime component, shifting other instances to keep the array contiguous.  
*Code lines:* 39  
*Contained by:* module galacticus_nodes  
*Modules used:* galacticus_error

**function:** node_component_formationtime_type  
*Description:* Returns the type for the formationTime component.  
*Code lines:* 8  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_formationtimecole2000_builder  
*Description:* Build a Cole2000 implementation of the formationTime component.  
*Code lines:* 19  
*Contained by:* module galacticus_nodes  
*Modules used:* fox_dom galacticus_error memory_management

**function:** node_component_formationtimecole2000_count  
*Description:* Return a count of the serialization of a Cole2000 implementation of the formationTime component.  
*Code lines:* 7  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_formationtimecole2000_deserialize_rates  
*Description:* Serialize rates of a Cole2000 implementation of the formationTime component.  
*Code lines:* 7  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_formationtimecole2000_deserialize_scales  
*Description:* Serialize scales of a Cole2000 implementation of the formationTime component.
19. Source Code Documentation

Code lines: 7  
Contained by: module galacticus_nodes

subroutine: node_component_formationtimecole2000_deserialize_values  
Description: Serialize values of a Cole2000 implementation of the formationTime component.  
Code lines: 7  
Contained by: module galacticus_nodes

subroutine: node_component_formationtimecole2000_destroy  
Description: Destroy a Cole2000 implementation of the formationTime component.  
Code lines: 7  
Contained by: module galacticus_nodes

Modules used: memory_management

subroutine: node_component_formationtimecole2000_dump  
Description: Dump the contents of a Cole2000 implementation of the formationTime component.  
Code lines: 16  
Contained by: module galacticus_nodes

Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_formationtimecole2000_dump_raw  
Description: Dump the contents of a Cole2000 implementation of the formationTime component in binary.  
Code lines: 8  
Contained by: module galacticus_nodes

subroutine: node_component_formationtimecole2000_dump_xml  
Description: Dump the contents of a Cole2000 implementation of the formationTime component to XML.  
Code lines: 10  
Contained by: module galacticus_nodes

subroutine: node_component_formationtimecole2000_initializer  
Description: Initialize a Cole2000 implementation of the formationTime component.  
Code lines: 8  
Contained by: module galacticus_nodes

Modules used: memory_management

function: node_component_formationtimecole2000_is_active  
Description: Return true if the Cole2000 implementation of the formationTime component is the active choice.  
Code lines: 6  
Contained by: module galacticus_nodes

subroutine: node_component_formationtimecole2000_name_from_index  
Description: Return the name of the property of given index for a Cole2000 implementation of the formationTime component.  
Code lines: 9  
Contained by: module galacticus_nodes

Modules used: iso_varying_string
**19.1. Program units**

**subroutine:** node_component_formationtimecole2000_ode_step_rates_init
*Description:* Initialize rates in a Cole2000 implementation of the formationTime component for an ODE solver step.
*Code lines:* 6
*Contained by:* module galacticus_nodes

**subroutine:** node_component_formationtimecole2000_odestepscalesinit
*Description:* Initialize scales in a Cole2000 implementation of the formationTime component for an ODE solver step.
*Code lines:* 5
*Contained by:* module galacticus_nodes

**subroutine:** node_component_formationtimecole2000_output_count
*Description:* Increment output property count for a Cole2000 implementation of the formationTime component.
*Code lines:* 9
*Contained by:* module galacticus_nodes

**subroutine:** node_component_formationtimecole2000_output_names
*Description:* Establish property names for a Cole2000 implementation of the formationTime component.
*Code lines:* 11
*Contained by:* module galacticus_nodes

**subroutine:** node_component_formationtimecole2000_read_raw
*Description:* Read the contents of a Cole2000 implementation of the formationTime component in binary.
*Code lines:* 9
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**subroutine:** node_component_formationtimecole2000_serialize_rates
*Description:* Serialize rates of a Cole2000 implementation of the formationTime component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** node_component_formationtimecole2000_serialize_scales
*Description:* Serialize scales of a Cole2000 implementation of the formationTime component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** node_component_formationtimecole2000_serialize_values
*Description:* Serialize values of a Cole2000 implementation of the formationTime component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**function:** node_component_formationtimecole2000_type
*Description:* Returns the type for the Cole2000 implementation of the formationTime component.
*Code lines:* 8
*Contained by:* module galacticus_nodes
subroutine: node_component_formationtimenull_builder
Description:建一个null实现的形成时间组件。
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom, galacticus_error, memory_management

function: node_component_formationtimenull_count
Description:返回null实现的形成时间组件的序列化计数。
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_formationtimenull_deserialize_rates
Description:序列化null实现的形成时间组件的速率。
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_formationtimenull_deserialize_scales
Description:序列化null实现的形成时间组件的标度。
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_formationtimenull_deserialize_values
Description:序列化null实现的形成时间组件的值。
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_formationtimenull_destroy
Description:销毁null实现的形成时间组件。
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_formationtimenull_dump
Description:打印null实现的形成时间组件的内容。
Code lines: 9
Contained by: module galacticus_nodes
Modules used: galacticus_display, iso_varying_string, string_handling

subroutine: node_component_formationtimenull_dump_raw
Description:打印null实现的形成时间组件的内容，以二进制形式。
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_formationtimenull_dump_xml
Description:打印null实现的形成时间组件的内容，以XML形式。
Code lines: 7
19.1. Program units

-- Contained by: module galacticus_nodes

**subroutine**: node_component_formationtimenull_initializor

*Description:* Initialize a null implementation of the formationTime component.

*Code lines:* 7

*Contained by:* module galacticus_nodes

*Modules used:* memory_management

**function**: node_component_formationtimenull_is_active

*Description:* Return true if the null implementation of the formationTime component is the active choice.

*Code lines:* 6

*Contained by:* module galacticus_nodes

**subroutine**: node_component_formationtimenull_name_from_index

*Description:* Return the name of the property of given index for a null implementation of the formationTime component.

*Code lines:* 9

*Contained by:* module galacticus_nodes

*Modules used:* iso_varying_string

**subroutine**: node_component_formationtimenull_ode_step_rates_init

*Description:* Initialize rates in a null implementation of the formationTime component for an ODE solver step.

*Code lines:* 6

*Contained by:* module galacticus_nodes

**subroutine**: node_component_formationtimenull_odestepscalesinit

*Description:* Initialize scales in a null implementation of the formationTime component for an ODE solver step.

*Code lines:* 5

*Contained by:* module galacticus_nodes

**subroutine**: node_component_formationtimenull_output_count

*Description:* Increment output property count for a null implementation of the formationTime component.

*Code lines:* 9

*Contained by:* module galacticus_nodes

**subroutine**: node_component_formationtimenull_output_names

*Description:* Establish property names for a null implementation of the formationTime component.

*Code lines:* 11

*Contained by:* module galacticus_nodes

**subroutine**: node_component_formationtimenull_read_raw

*Description:* Read the contents of a null implementation of the formationTime component in binary.

*Code lines:* 8

*Contained by:* module galacticus_nodes

*Modules used:* memory_management
subroutine: node_component_formationtimenull_serialize_rates
Description: Serialize rates of a null implementation of the formationTime component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_formationtimenull_serialize_scales
Description: Serialize scales of a null implementation of the formationTime component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_formationtimenull_serialize_values
Description: Serialize values of a null implementation of the formationTime component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_formationtimenull_type
Description: Returns the type for the null implementation of the formationTime component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_generic_destroy
Description: Destroy a generic tree node component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_generic_type
Description: Returns the name of a generic tree node component.
Code lines: 8
Contained by: module galacticus_nodes

function: node_component_host_node
Description: Return the host tree node of a tree node component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_hosthistory_builder
Description: Build a generic hostHistory component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error

subroutine: node_component_hosthistory_destroy
Description: Destroys a hostHistory component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_hosthistory_dump
Description: Dump the contents of a generic hostHistory component.
19.1. Program units

subroutine: node_component_hosthistory_initializor
Description: Initialize a generic hostHistory component.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string

subroutine: node_component_hosthistory_move
Description: Move instances of the hostHistory component, from one node to another.
Code lines: 67
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_hosthistory_output
Description: Output properties for a hostHistory component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_hosthistory_output_count
Description: Increment the count of properties to output for a generic hostHistory component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_hosthistory_output_names
Description: Establish property names for a generic hostHistory component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_hosthistory_remove
Description: Removes an instance of the hostHistory component, shifting other instances to keep the array contiguous.
Code lines: 39
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: node_component_hosthistory_type
Description: Returns the type for the hostHistory component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorynull_builder
Description: Build a null implementation of the hostHistory component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management
function: node_component_hosthistorynull_count
Description: Return a count of the serialization of a null implementation of the hostHistory component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorynull_deserialize_rates
Description: Serialize rates of a null implementation of the hostHistory component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorynull_deserialize_scales
Description: Serialize scales of a null implementation of the hostHistory component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorynull_deserialize_values
Description: Serialize values of a null implementation of the hostHistory component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorynull_destroy
Description: Destroy a null implementation of the hostHistory component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_hosthistorynull_dump
Description: Dump the contents of a null implementation of the hostHistory component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_hosthistorynull_dump_raw
Description: Dump the contents of a null implementation of the hostHistory component in binary.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorynull_dump_xml
Description: Dump the contents of a null implementation of the hostHistory component to XML.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorynull_initializor
Description: Initialize a null implementation of the hostHistory component.
Code lines: 7
Contained by: module galacticus_nodes
19.1. Program units

Modules used: memory_management

function: node_component_hosthistorynull_is_active
Description: Return true if the null implementation of the hostHistory component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorynull_name_from_index
Description: Return the name of the property of given index for a null implementation of the hostHistory component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_hosthistorynull_ode_step_rates_init
Description: Initialize rates in a null implementation of the hostHistory component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorynull_odestepscalesinit
Description: Initialize scales in a null implementation of the hostHistory component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorynull_output_count
Description: Increment output property count for a null implementation of the hostHistory component.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorynull_output_names
Description: Establish property names for a null implementation of the hostHistory component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorynull_read_raw
Description: Read the contents of a null implementation of the hostHistory component in binary.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_hosthistorynull_serialize_rates
Description: Serialize rates of a null implementation of the hostHistory component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorynull_serialize_scales
Description: Serialize scales of a null implementation of the hostHistory component.
19. Source Code Documentation

subroutine: node_component_hosthistorynull_serialize_values
Description: Serialize values of a null implementation of the hostHistory component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_hosthistorynull_type
Description: Returns the type for the null implementation of the hostHistory component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorystandard_builder
Description: Build a standard implementation of the hostHistory component.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_hosthistorystandard_count
Description: Return a count of the serialization of a standard implementation of the hostHistory component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorystandard_deserialize_rates
Description: Serialize rates of a standard implementation of the hostHistory component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorystandard_deserialize_scales
Description: Serialize scales of a standard implementation of the hostHistory component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorystandard_deserialize_values
Description: Serialize values of a standard implementation of the hostHistory component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorystandard_destroy
Description: Destroy a standard implementation of the hostHistory component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_hosthistorystandard_dump
Description: Dump the contents of a standard implementation of the hostHistory component.
subroutine: node_component_hosthistorystandard_dump_raw
Description: Dump the contents of a standard implementation of the hostHistory component in binary.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorystandard_dump_xml
Description: Dump the contents of a standard implementation of the hostHistory component to XML.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorystandard_initializer
Description: Initialize a standard implementation of the hostHistory component.
Code lines: 8
Contained by: module galacticus_nodes

function: node_component_hosthistorystandard_is_active
Description: Return true if the standard implementation of the hostHistory component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorystandard_name_from_index
Description: Return the name of the property of given index for a standard implementation of the hostHistory component.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorystandard_ode_step_rates_init
Description: Initialize rates in a standard implementation of the hostHistory component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorystandard_ode_stepscalesinit
Description: Initialize scales in a standard implementation of the hostHistory component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_hosthistorystandard_output_count
Description: Increment output property count for a standard implementation of the hostHistory component.
Code lines: 10
19. Source Code Documentation

Contained by: module galacticus_nodes

**subroutine: node_component_hosthistorystandard_output_names**
*Description:* Establish property names for a standard implementation of the hostHistory component.
*Code lines:* 24
*Contained by:* module galacticus_nodes

**subroutine: node_component_hosthistorystandard_read_raw**
*Description:* Read the contents of a standard implementation of the hostHistory component in binary.
*Code lines:* 9
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**subroutine: node_component_hosthistorystandard_serialize_rates**
*Description:* Serialize rates of a standard implementation of the hostHistory component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine: node_component_hosthistorystandard_serialize_scales**
*Description:* Serialize scales of a standard implementation of the hostHistory component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine: node_component_hosthistorystandard_serialize_values**
*Description:* Serialize values of a standard implementation of the hostHistory component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**function: node_component_hosthistorystandard_type**
*Description:* Returns the type for the standard implementation of the hostHistory component.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine: node_component_hot_halo_standard_mass_removal_rate**
*Description:* Handle instances of mass removal from the standard hot halo component class. For the standard hot halo component type, we do nothing.
*Code lines:* 9
*Contained by:* module galacticus_nodes
*Modules used:* galacticus_error

**subroutine: node_component_hot_halo_builder**
*Description:* Build a generic hotHalo component.
*Code lines:* 10
*Contained by:* module galacticus_nodes
*Modules used:* fox_dom galacticus_error

**subroutine: node_component_hot_halo_destroy**
*Description:* Destroys a hotHalo component.
*Code lines:* 7
19.1. Program units

**Contained by:** module galacticus_nodes

**subroutine:** node_component_hothalo_dump
- **Description:** Dump the contents of a generic hotHalo component.
- **Code lines:** 10
- **Contained by:** module galacticus_nodes
- **Modules used:** galacticus_display, iso_varying_string

**subroutine:** node_component_hothalo_initializor
- **Description:** Initialize a generic hotHalo component.
- **Code lines:** 8
- **Contained by:** module galacticus_nodes
- **Modules used:** galacticus_error

**subroutine:** node_component_hothalo_move
- **Description:** Move instances of the hotHalo component, from one node to another.
- **Code lines:** 109
- **Contained by:** module galacticus_nodes
- **Modules used:** galacticus_error

**subroutine:** node_component_hothalo_output
- **Description:** Output properties for a hotHalo component.
- **Code lines:** 48
- **Contained by:** module galacticus_nodes

**subroutine:** node_component_hothalo_output_count
- **Description:** Increment the count of properties to output for a generic hotHalo component.
- **Code lines:** 13
- **Contained by:** module galacticus_nodes

**subroutine:** node_component_hothalo_output_names
- **Description:** Establish property names for a generic hotHalo component.
- **Code lines:** 15
- **Contained by:** module galacticus_nodes

**subroutine:** node_component_hothalo_remove
- **Description:** Removes an instance of the hotHalo component, shifting other instances to keep the array contiguous.
- **Code lines:** 63
- **Contained by:** module galacticus_nodes
- **Modules used:** galacticus_error

**function:** node_component_hothalo_type
- **Description:** Returns the type for the hotHalo component.
- **Code lines:** 8
- **Contained by:** module galacticus_nodes

**subroutine:** node_component_hothalocoldmode_builder
- **Description:** Build a coldMode implementation of the hotHalo component.
19. Source Code Documentation

**function:** node_component_hothalocoldmode_count
**Description:** Return a count of the serialization of a coldMode implementation of the hotHalo component.
**Code lines:** 9
**Contained by:** module galacticus_nodes

```
contained by: module galacticus_nodes
modules used: fox_dom memory_management galacticus_error
```

**subroutine:** node_component_hothalocoldmode_deserialize_rates
**Description:** Serialize rates of a coldMode implementation of the hotHalo component.
**Code lines:** 21
**Contained by:** module galacticus_nodes

**subroutine:** node_component_hothalocoldmode_deserialize_scales
**Description:** Serialize scales of a coldMode implementation of the hotHalo component.
**Code lines:** 21
**Contained by:** module galacticus_nodes

**subroutine:** node_component_hothalocoldmode_deserialize_values
**Description:** Serialize values of a coldMode implementation of the hotHalo component.
**Code lines:** 21
**Contained by:** module galacticus_nodes

**subroutine:** node_component_hothalocoldmode_destroy
**Description:** Destroy a coldMode implementation of the hotHalo component.
**Code lines:** 10
**Contained by:** module galacticus_nodes
**Modules used:** memory_management

```
subroutine: node_component_hothalocoldmode_destroy
contained by: module galacticus_nodes
modules used: memory_management
```

**subroutine:** node_component_hothalocoldmode_dump
**Description:** Dump the contents of a coldMode implementation of the hotHalo component.
**Code lines:** 24
**Contained by:** module galacticus_nodes
**Modules used:** galacticus_display iso_varying_string string_handling

```
subroutine: node_component_hothalocoldmode_dump
contained by: module galacticus_nodes
modules used: galacticus_display iso_varying_string string_handling
```

**subroutine:** node_component_hothalocoldmode_dump_raw
**Description:** Dump the contents of a coldMode implementation of the hotHalo component in binary.
**Code lines:** 11
**Contained by:** module galacticus_nodes

**subroutine:** node_component_hothalocoldmode_dump_xml
**Description:** Dump the contents of a coldMode implementation of the hotHalo component to XML.
**Code lines:** 14
**Contained by:** module galacticus_nodes
subroutine: node_component_hothalocoldmode_initializor
Description: Initialize a coldMode implementation of the hotHalo component.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_hothalocoldmode_is_active
Description: Return true if the coldMode implementation of the hotHalo component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_hothalocoldmode_name_from_index
Description: Return the name of the property of given index for a coldMode implementation of the hotHalo component.
Code lines: 26
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_hothalocoldmode_ode_step_rates_init
Description: Initialize rates in a coldMode implementation of the hotHalo component for an ODE solver step.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_hothalocoldmode_odestepscalesinit
Description: Initialize scales in a coldMode implementation of the hotHalo component for an ODE solver step.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_hothalocoldmode_output_count
Description: Increment output property count for a coldMode implementation of the hotHalo component.
Code lines: 14
Contained by: module galacticus_nodes

subroutine: node_component_hothalocoldmode_output_names
Description: Establish property names for a coldMode implementation of the hotHalo component.
Code lines: 41
Contained by: module galacticus_nodes

subroutine: node_component_hothalocoldmode_read_raw
Description: Read the contents of a coldMode implementation of the hotHalo component in binary.
Code lines: 12
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_hothalocoldmode_serialize_rates
Description: Serialize rates of a coldMode implementation of the hotHalo component.
19. Source Code Documentation

Code lines: 21
Contained by: module galacticus_nodes

**subroutine**: node_component_hothalocoldmode_serialize_scales
*Description*: Serialize scales of a coldMode implementation of the hotHalo component.
*Code lines*: 21
*Contained by*: module galacticus_nodes

**subroutine**: node_component_hothalocoldmode_serialize_values
*Description*: Serialize values of a coldMode implementation of the hotHalo component.
*Code lines*: 21
*Contained by*: module galacticus_nodes

**function**: node_component_hothalocoldmode_type
*Description*: Returns the type for the coldMode implementation of the hotHalo component.
*Code lines*: 8
*Contained by*: module galacticus_nodes

**subroutine**: node_component_hothalonull_builder
*Description*: Build a null implementation of the hotHalo component.
*Code lines*: 10
*Contained by*: module galacticus_nodes
*Modules used*: fox_dom galacticus_error memory_management

**function**: node_component_hothalonull_count
*Description*: Return a count of the serialization of a null implementation of the hotHalo component.
*Code lines*: 7
*Contained by*: module galacticus_nodes

**subroutine**: node_component_hothalonull_deserialize_rates
*Description*: Serialize rates of a null implementation of the hotHalo component.
*Code lines*: 7
*Contained by*: module galacticus_nodes

**subroutine**: node_component_hothalonull_deserialize_scales
*Description*: Serialize scales of a null implementation of the hotHalo component.
*Code lines*: 7
*Contained by*: module galacticus_nodes

**subroutine**: node_component_hothalonull_deserialize_values
*Description*: Serialize values of a null implementation of the hotHalo component.
*Code lines*: 7
*Contained by*: module galacticus_nodes

**subroutine**: node_component_hothalonull_destroy
*Description*: Destroy a null implementation of the hotHalo component.
*Code lines*: 7
*Contained by*: module galacticus_nodes
19.1. Program units

*Modules used:* memory_management

**subroutine:** node_component_hothalonull_dump
*Description:* Dump the contents of a null implementation of the hotHalo component.
*Code lines:* 9
*Contained by:* module galacticus_nodes
*Modules used:* galacticus_display

**subroutine:** node_component_hothalonull_dump_raw
*Description:* Dump the contents of a null implementation of the hotHalo component in binary.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** node_component_hothalonull_dump_xml
*Description:* Dump the contents of a null implementation of the hotHalo component to XML.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** node_component_hothalonull_initializor
*Description:* Initialize a null implementation of the hotHalo component.
*Code lines:* 7
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**function:** node_component_hothalonull_is_active
*Description:* Return true if the null implementation of the hotHalo component is the active choice.
*Code lines:* 6
*Contained by:* module galacticus_nodes

**subroutine:** node_component_hothalonull_name_from_index
*Description:* Return the name of the property of given index for a null implementation of the hotHalo component.
*Code lines:* 9
*Contained by:* module galacticus_nodes
*Modules used:* iso_varying_string

**subroutine:** node_component_hothalonull_ode_step_rates_init
*Description:* Initialize rates in a null implementation of the hotHalo component for an ODE solver step.
*Code lines:* 6
*Contained by:* module galacticus_nodes

**subroutine:** node_component_hothalonull_odestepscalesinit
*Description:* Initialize scales in a null implementation of the hotHalo component for an ODE solver step.
*Code lines:* 5
*Contained by:* module galacticus_nodes

**subroutine:** node_component_hothalonull_output_count
*Description:* Increment output property count for a null implementation of the hotHalo component.
19. Source Code Documentation

Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_hothalonull_output_names
Description: Establish property names for a null implementation of the hotHalo component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_hothalonull_read_raw
Description: Read the contents of a null implementation of the hotHalo component in binary.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_hothalonull_serialize_rates
Description: Serialize rates of a null implementation of the hotHalo component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_hothalonull_serialize_scales
Description: Serialize scales of a null implementation of the hotHalo component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_hothalonull_serialize_values
Description: Serialize values of a null implementation of the hotHalo component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_hothalonull_type
Description: Returns the type for the null implementation of the hotHalo component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_hothalooutflowtracking_builder
Description: Build a outflowTracking implementation of the hotHalo component.
Code lines: 26
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_hothalooutflowtracking_count
Description: Return a count of the serialization of a outflowTracking implementation of the hotHalo component.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_hothalooutflowtracking_deserialize_rates
Description: Serialize rates of a outflowTracking implementation of the hotHalo component.
19.1. Program units

subroutine: node_component_hothalooutflowtracking_deserialize_scales
Description: Serialize scales of a outflowTracking implementation of the hotHalo component.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: node_component_hothalooutflowtracking_deserialize_values
Description: Serialize values of a outflowTracking implementation of the hotHalo component.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: node_component_hothalooutflowtracking_destroy
Description: Destroy a outflowTracking implementation of the hotHalo component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_hothalooutflowtracking_dump
Description: Dump the contents of a outflowTracking implementation of the hotHalo component.
Code lines: 21
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_hothalooutflowtracking_dump_raw
Description: Dump the contents of a outflowTracking implementation of the hotHalo component in binary.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_hothalooutflowtracking_dump_xml
Description: Dump the contents of a outflowTracking implementation of the hotHalo component to XML.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_hothalooutflowtracking_initializor
Description: Initialize a outflowTracking implementation of the hotHalo component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_hothalooutflowtracking_is_active
Description: Return true if the outflowTracking implementation of the hotHalo component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_hothalooutflowtracking_name_from_index
**19. Source Code Documentation**

*Description:* Return the name of the property of given index for a outflowTracking implementation of the hotHalo component.

*Code lines:* 21

*Contained by:* module `galacticus_nodes`

*Modules used:* `iso_varying_string`

**subroutine:** `node_component_hothalooutflowtracking_ode_step_rates_init`

*Description:* Initialize rates in a outflowTracking implementation of the hotHalo component for an ODE solver step.

*Code lines:* 9

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_hothalooutflowtracking_odestepscalesinit`

*Description:* Initialize scales in a outflowTracking implementation of the hotHalo component for an ODE solver step.

*Code lines:* 8

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_hothalooutflowtracking_output_count`

*Description:* Increment output property count for a outflowTracking implementation of the hotHalo component.

*Code lines:* 14

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_hothalooutflowtracking_output_names`

*Description:* Establish property names for a outflowTracking implementation of the hotHalo component.

*Code lines:* 28

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_hothalooutflowtracking_read_raw`

*Description:* Read the contents of a outflowTracking implementation of the hotHalo component in binary.

*Code lines:* 11

*Contained by:* module `galacticus_nodes`

*Modules used:* `memory_management`

**subroutine:** `node_component_hothalooutflowtracking_serialize_rates`

*Description:* Serialize rates of a outflowTracking implementation of the hotHalo component.

*Code lines:* 19

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_hothalooutflowtracking_serialize_scales`

*Description:* Serialize scales of a outflowTracking implementation of the hotHalo component.

*Code lines:* 19

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_hothalooutflowtracking_serialize_values`

*Description:* Serialize values of a outflowTracking implementation of the hotHalo component.

*Code lines:* 19

*Contained by:* module `galacticus_nodes`
function: node_component_hothalooutflowtracking_type
Description: Returns the type for the outflowTracking implementation of the hotHalo component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_builder
Description: Build a standard implementation of the hotHalo component.
Code lines: 85
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_hothalostandard_count
Description: Return a count of the serialization of a standard implementation of the hotHalo component.
Code lines: 12
Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_deserialize_rates
Description: Serialize rates of a standard implementation of the hotHalo component.
Code lines: 35
Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_deserialize_scales
Description: Serialize scales of a standard implementation of the hotHalo component.
Code lines: 35
Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_deserialize_values
Description: Serialize values of a standard implementation of the hotHalo component.
Code lines: 35
Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_destroy
Description: Destroy a standard implementation of the hotHalo component.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_hothalostandard_dump
Description: Dump the contents of a standard implementation of the hotHalo component.
Code lines: 53
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_hothalostandard_dump_raw
Description: Dump the contents of a standard implementation of the hotHalo component in binary.
Code lines: 19
19. Source Code Documentation

Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_dump_xml
Description: Dump the contents of a standard implementation of the hotHalo component to XML.
Code lines: 31
Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_initializor
Description: Initialize a standard implementation of the hotHalo component.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_hothalostandard_is_active
Description: Return true if the standard implementation of the hotHalo component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_name_from_index
Description: Return the name of the property of given index for a standard implementation of the hotHalo component.
Code lines: 64
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_hothalostandard_ode_step_rates_init
Description: Initialize rates in a standard implementation of the hotHalo component for an ODE solver step.
Code lines: 17
Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_odestepscalesinit
Description: Initialize scales in a standard implementation of the hotHalo component for an ODE solver step.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_output_count
Description: Increment output property count for a standard implementation of the hotHalo component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_output_names
Description: Establish property names for a standard implementation of the hotHalo component.
Code lines: 94
Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_read_raw
Description: Read the contents of a standard implementation of the hotHalo component in binary.
19.1. Program units

Code lines: 20  
Contained by: module galacticus_nodes  
Modules used: memory_management

subroutine: node_component_hothalostandard_serialize_rates  
Description: Serialize rates of a standard implementation of the hotHalo component.  
Code lines: 35  
Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_serialize_scales  
Description: Serialize scales of a standard implementation of the hotHalo component.  
Code lines: 35  
Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_serialize_values  
Description: Serialize values of a standard implementation of the hotHalo component.  
Code lines: 35  
Contained by: module galacticus_nodes

function: node_component_hothalostandard_type  
Description: Returns the type for the standard implementation of the hotHalo component.  
Code lines: 8  
Contained by: module galacticus_nodes

subroutine: node_component_hothaloverysimple_builder  
Description: Build a verySimple implementation of the hotHalo component.  
Code lines: 19  
Contained by: module galacticus_nodes  
Modules used: fox_dom galacticus_error memory_management

function: node_component_hothaloverysimple_count  
Description: Return a count of the serialization of a verySimple implementation of the hotHalo component.  
Code lines: 8  
Contained by: module galacticus_nodes

subroutine: node_component_hothaloverysimple_deserialize_rates  
Description: Serialize rates of a verySimple implementation of the hotHalo component.  
Code lines: 11  
Contained by: module galacticus_nodes

subroutine: node_component_hothaloverysimple_deserialize_scales  
Description: Serialize scales of a verySimple implementation of the hotHalo component.  
Code lines: 11  
Contained by: module galacticus_nodes

subroutine: node_component_hothaloverysimple_deserialize_values  
Description: Serialize values of a verySimple implementation of the hotHalo component.
Code lines: 11  
Contained by: module galacticus_nodes

**subroutine: node_component_hothaloverysimple_destroy**
*Description:* Destroy a verySimple implementation of the hotHalo component.
*Code lines:* 7  
*Contained by:* module galacticus_nodes  
*Modules used:* memory_management

**subroutine: node_component_hothaloverysimple_dump**
*Description:* Dump the contents of a verySimple implementation of the hotHalo component.
*Code lines:* 16  
*Contained by:* module galacticus_nodes  
*Modules used:* galacticus_display iso_varying_string string_handling

**subroutine: node_component_hothaloverysimple_dump_raw**
*Description:* Dump the contents of a verySimple implementation of the hotHalo component in binary.
*Code lines:* 8  
*Contained by:* module galacticus_nodes

**subroutine: node_component_hothaloverysimple_dump_xml**
*Description:* Dump the contents of a verySimple implementation of the hotHalo component to XML.
*Code lines:* 13  
*Contained by:* module galacticus_nodes

**subroutine: node_component_hothaloverysimple_initializor**
*Description:* Initialize a verySimple implementation of the hotHalo component.
*Code lines:* 8  
*Contained by:* module galacticus_nodes  
*Modules used:* memory_management

**function: node_component_hothaloverysimple_is_active**
*Description:* Return true if the verySimple implementation of the hotHalo component is the active choice.
*Code lines:* 6  
*Contained by:* module galacticus_nodes

**subroutine: node_component_hothaloverysimple_name_from_index**
*Description:* Return the name of the property of given index for a verySimple implementation of the hotHalo component.
*Code lines:* 14  
*Contained by:* module galacticus_nodes  
*Modules used:* iso_varying_string

**subroutine: node_component_hothaloverysimple_ode_step_rates_init**
*Description:* Initialize rates in a verySimple implementation of the hotHalo component for an ODE solver step.
*Code lines:* 7  
*Contained by:* module galacticus_nodes
19.1. Program units

**subroutine**: node_component_hothaloverysimple_odestepscalesinit
*Description:* Initialize scales in a verySimple implementation of the hotHalo component for an ODE solver step.
*Code lines:* 6
*Contained by:* module galacticus_nodes

**subroutine**: node_component_hothaloverysimple_output_count
*Description:* Increment output property count for a verySimple implementation of the hotHalo component.
*Code lines:* 10
*Contained by:* module galacticus_nodes

**subroutine**: node_component_hothaloverysimple_output_names
*Description:* Establish property names for a verySimple implementation of the hotHalo component.
*Code lines:* 24
*Contained by:* module galacticus_nodes

**subroutine**: node_component_hothaloverysimple_read_raw
*Description:* Read the contents of a verySimple implementation of the hotHalo component in binary.
*Code lines:* 9
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**subroutine**: node_component_hothaloverysimple_serialize_rates
*Description:* Serialize rates of a verySimple implementation of the hotHalo component.
*Code lines:* 11
*Contained by:* module galacticus_nodes

**subroutine**: node_component_hothaloverysimple_serialize_scales
*Description:* Serialize scales of a verySimple implementation of the hotHalo component.
*Code lines:* 11
*Contained by:* module galacticus_nodes

**subroutine**: node_component_hothaloverysimple_serialize_values
*Description:* Serialize values of a verySimple implementation of the hotHalo component.
*Code lines:* 11
*Contained by:* module galacticus_nodes

**function**: node_component_hothaloverysimple_type
*Description:* Returns the type for the verySimple implementation of the hotHalo component.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine**: node_component_indices_builder
*Description:* Build a generic indices component.
*Code lines:* 10
*Contained by:* module galacticus_nodes
*Modules used:* fox_dom galacticus_error
subroutine: node_component_indices_destroy
Description: Destroys a indices component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_indices_dump
Description: Dump the contents of a generic indices component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: galacticus_display, iso_varying_string

subroutine: node_component_indices_initializor
Description: Initialize a generic indices component.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_indices_move
Description: Move instances of the indices component, from one node to another.
Code lines: 67
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_indices_output
Description: Output properties for a indices component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_indices_output_count
Description: Increment the count of properties to output for a generic indices component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_indices_output_names
Description: Establish property names for a generic indices component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_indices_remove
Description: Removes an instance of the indices component, shifting other instances to keep the array contiguous.
Code lines: 39
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: node_component_indices_type
Description: Returns the type for the indices component.
Code lines: 8
19.1. Program units

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_indicesnull_builder`
*Description:* Build a null implementation of the indices component.
*Code lines:* 10
*Contained by:* module `galacticus_nodes`
*Modules used:* `fox_dom` `galacticus_error` `memory_management`

**function:** `node_component_indicesnull_count`
*Description:* Return a count of the serialization of a null implementation of the indices component.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_indicesnull_deserialize_rates`
*Description:* Serialize rates of a null implementation of the indices component.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_indicesnull_deserialize_scales`
*Description:* Serialize scales of a null implementation of the indices component.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_indicesnull_deserialize_values`
*Description:* Serialize values of a null implementation of the indices component.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_indicesnull_destroy`
*Description:* Destroy a null implementation of the indices component.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`
*Modules used:* `memory_management`

**subroutine:** `node_component_indicesnull_dump`
*Description:* Dump the contents of a null implementation of the indices component.
*Code lines:* 9
*Contained by:* module `galacticus_nodes`
*Modules used:* `galacticus_display` `iso_varying_string` `string_handling`

**subroutine:** `node_component_indicesnull_dump_raw`
*Description:* Dump the contents of a null implementation of the indices component in binary.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_indicesnull_dump_xml`
*Description:* Dump the contents of a null implementation of the indices component to XML.
subroutine: node_component_indicesnull_initializor
Description: Initialize a null implementation of the indices component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_indicesnull_is_active
Description: Return true if the null implementation of the indices component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_indicesnull_name_from_index
Description: Return the name of the property of given index for a null implementation of the indices component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_indicesnull_ode_step_rates_init
Description: Initialize rates in a null implementation of the indices component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_indicesnull_odestepscalesinit
Description: Initialize scales in a null implementation of the indices component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_indicesnull_output_count
Description: Increment output property count for a null implementation of the indices component.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_indicesnull_output_names
Description: Establish property names for a null implementation of the indices component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_indicesnull_read_raw
Description: Read the contents of a null implementation of the indices component in binary.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_indicesnull_serialize_rates
Description: Serialize rates of a null implementation of the indices component.
19.1. Program units

subroutine: node_component_indicesnull_serialize_scales
Description: Serialize scales of a null implementation of the indices component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_indicesnull_serialize_values
Description: Serialize values of a null implementation of the indices component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_indicesnull_type
Description: Returns the type for the null implementation of the indices component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_indicesstandard_builder
Description: Build a standard implementation of the indices component.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_indicesstandard_count
Description: Return a count of the serialization of a standard implementation of the indices component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_indicesstandard_deserialize_rates
Description: Serialize rates of a standard implementation of the indices component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_indicesstandard_deserialize_scales
Description: Serialize scales of a standard implementation of the indices component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_indicesstandard_deserialize_values
Description: Serialize values of a standard implementation of the indices component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_indicesstandard_destroy
Description: Destroy a standard implementation of the indices component.
Code lines: 7
Contained by: module galacticus_nodes
19. Source Code Documentation

Modules used: memory_management

subroutine: node_component_indicesstandard_dump
Description: Dump the contents of a standard implementation of the indices component.
Code lines: 16
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_indicesstandard_dump_raw
Description: Dump the contents of a standard implementation of the indices component in binary.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_indicesstandard_dump_xml
Description: Dump the contents of a standard implementation of the indices component to XML.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_indicesstandard_initializor
Description: Initialize a standard implementation of the indices component.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_indicesstandard_is_active
Description: Return true if the standard implementation of the indices component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_indicesstandard_name_from_index
Description: Return the name of the property of given index for a standard implementation of the indices component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_indicesstandard_ode_step_rates_init
Description: Initialize rates in a standard implementation of the indices component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_indicesstandard_odestepscalesinit
Description: Initialize scales in a standard implementation of the indices component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_indicesstandard_output_count
19.1. Program units

Description: Increment output property count for a standard implementation of the indices component.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_indicesstandard_output_names
Description: Establish property names for a standard implementation of the indices component.
Code lines: 24
Contained by: module galacticus_nodes

subroutine: node_component_indicesstandard_read_raw
Description: Read the contents of a standard implementation of the indices component in binary.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_indicesstandard_serialize_rates
Description: Serialize rates of a standard implementation of the indices component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_indicesstandard_serialize_scales
Description: Serialize scales of a standard implementation of the indices component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_indicesstandard_serialize_values
Description: Serialize values of a standard implementation of the indices component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_indicesstandard_type
Description: Returns the type for the standard implementation of the indices component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_interoutput_builder
Description: Build a generic interOutput component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error

subroutine: node_component_interoutput_destroy
Description: Destroys a interOutput component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_interoutput_dump
Description: Dump the contents of a generic interOutput component.
Code lines: 10
19. Source Code Documentation

```
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string

subroutine: node_component_interoutput_initializor
  Description: Initialize a generic interOutput component.
  Code lines: 8
  Contained by: module galacticus_nodes
  Modules used: galacticus_error

subroutine: node_component_interoutput_move
  Description: Move instances of the interOutput component, from one node to another.
  Code lines: 67
  Contained by: module galacticus_nodes
  Modules used: galacticus_error

subroutine: node_component_interoutput_output
  Description: Output properties for a interOutput component.
  Code lines: 17
  Contained by: module galacticus_nodes

subroutine: node_component_interoutput_output_count
  Description: Increment the count of properties to output for a generic interOutput component.
  Code lines: 13
  Contained by: module galacticus_nodes

subroutine: node_component_interoutput_output_names
  Description: Establish property names for a generic interOutput component.
  Code lines: 15
  Contained by: module galacticus_nodes

subroutine: node_component_interoutput_remove
  Description: Removes an instance of the interOutput component, shifting other instances to keep the array contiguous.
  Code lines: 39
  Contained by: module galacticus_nodes
  Modules used: galacticus_error

function: node_component_interoutput_type
  Description: Returns the type for the interOutput component.
  Code lines: 8
  Contained by: module galacticus_nodes

subroutine: node_component_interoutputnull_builder
  Description: Build a null implementation of the interOutput component.
  Code lines: 10
  Contained by: module galacticus_nodes
  Modules used: fox_dom memory_management galacticus_error
```
function: node_component_interoutputnull_count
Description: Return a count of the serialization of a null implementation of the interOutput component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_interoutputnull_deserialize_rates
Description: Serialize rates of a null implementation of the interOutput component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_interoutputnull_deserialize_scales
Description: Serialize scales of a null implementation of the interOutput component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_interoutputnull_deserialize_values
Description: Serialize values of a null implementation of the interOutput component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_interoutputnull_destroy
Description: Destroy a null implementation of the interOutput component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_interoutputnull_dump
Description: Dump the contents of a null implementation of the interOutput component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_interoutputnull_dump_raw
Description: Dump the contents of a null implementation of the interOutput component in binary.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_interoutputnull_dump_xml
Description: Dump the contents of a null implementation of the interOutput component to XML.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_interoutputnull_initializor
Description: Initialize a null implementation of the interOutput component.
Code lines: 7
Contained by: module galacticus_nodes
19. Source Code Documentation

Modules used: memory_management

function: node_component_interoutputnull_is_active
Description: Return true if the null implementation of the interOutput component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_interoutputnull_name_from_index
Description: Return the name of the property of given index for a null implementation of the interOutput component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_interoutputnull_ode_step_rates_init
Description: Initialize rates in a null implementation of the interOutput component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_interoutputnull_odestepsscalesinit
Description: Initialize scales in a null implementation of the interOutput component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_interoutputnull_output_count
Description: Increment output property count for a null implementation of the interOutput component.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_interoutputnull_output_names
Description: Establish property names for a null implementation of the interOutput component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_interoutputnull_read_raw
Description: Read the contents of a null implementation of the interOutput component in binary.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_interoutputnull_serialize_rates
Description: Serialize rates of a null implementation of the interOutput component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_interoutputnull_serialize_scales
Description: Serialize scales of a null implementation of the interOutput component.
19.1. Program units

subroutine: node_component_interoutputnull_serialize_values
Description: Serialize values of a null implementation of the interOutput component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_interoutputnull_type
Description: Returns the type for the null implementation of the interOutput component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_interoutputstandard_builder
Description: Build a standard implementation of the interOutput component.
Code lines: 25
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_interoutputstandard_count
Description: Return a count of the serialization of a standard implementation of the interOutput component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_interoutputstandard_deserialize_rates
Description: Serialize rates of a standard implementation of the interOutput component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_interoutputstandard_deserialize_scales
Description: Serialize scales of a standard implementation of the interOutput component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_interoutputstandard_deserialize_values
Description: Serialize values of a standard implementation of the interOutput component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_interoutputstandard_destroy
Description: Destroy a standard implementation of the interOutput component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_interoutputstandard_dump
Description: Dump the contents of a standard implementation of the interOutput component.
subroutine: node_component_interoutputstandard_dump_raw
  Description: Dump the contents of a standard implementation of the interOutput component in binary.
  Code lines: 9
  Contained by: module galacticus_nodes

subroutine: node_component_interoutputstandard_dump_xml
  Description: Dump the contents of a standard implementation of the interOutput component to XML.
  Code lines: 11
  Contained by: module galacticus_nodes

subroutine: node_component_interoutputstandard_initializor
  Description: Initialize a standard implementation of the interOutput component.
  Code lines: 9
  Contained by: module galacticus_nodes
  Modules used: memory_management

function: node_component_interoutputstandard_is_active
  Description: Return true if the standard implementation of the interOutput component is the active choice.
  Code lines: 6
  Contained by: module galacticus_nodes

subroutine: node_component_interoutputstandard_name_from_index
  Description: Return the name of the property of given index for a standard implementation of the interOutput component.
  Code lines: 19
  Contained by: module galacticus_nodes
  Modules used: iso_varying_string

subroutine: node_component_interoutputstandard_ode_step_rates_init
  Description: Initialize rates in a standard implementation of the interOutput component for an ODE solver step.
  Code lines: 8
  Contained by: module galacticus_nodes

subroutine: node_component_interoutputstandard_odestepscalesinit
  Description: Initialize scales in a standard implementation of the interOutput component for an ODE solver step.
  Code lines: 7
  Contained by: module galacticus_nodes

subroutine: node_component_interoutputstandard_output_count
  Description: Increment output property count for a standard implementation of the interOutput component.
  Code lines: 10

19. Source Code Documentation
19.1. Program units

**Contained by:** module galacticus_nodes

**subroutine:** node_component_interoutputstandard_output_names
*Description:* Establish property names for a standard implementation of the interOutput component.
*Code lines:* 37
**Contained by:** module galacticus_nodes

**subroutine:** node_component_interoutputstandard_read_raw
*Description:* Read the contents of a standard implementation of the interOutput component in binary.
*Code lines:* 10
**Contained by:** module galacticus_nodes
**Modules used:** memory_management

**subroutine:** node_component_interoutputstandard_serialize_rates
*Description:* Serialize rates of a standard implementation of the interOutput component.
*Code lines:* 13
**Contained by:** module galacticus_nodes

**subroutine:** node_component_interoutputstandard_serialize_scales
*Description:* Serialize scales of a standard implementation of the interOutput component.
*Code lines:* 13
**Contained by:** module galacticus_nodes

**subroutine:** node_component_interoutputstandard_serialize_values
*Description:* Serialize values of a standard implementation of the interOutput component.
*Code lines:* 13
**Contained by:** module galacticus_nodes

**function:** node_component_interoutputstandard_type
*Description:* Returns the type for the standard implementation of the interOutput component.
*Code lines:* 8
**Contained by:** module galacticus_nodes

**subroutine:** node_component_massflowstatistics_builder
*Description:* Build a generic massFlowStatistics component.
*Code lines:* 10
**Contained by:** module galacticus_nodes
**Modules used:** fox_dom galacticus_error

**subroutine:** node_component_massflowstatistics_destroy
*Description:* Destroys a massFlowStatistics component.
*Code lines:* 7
**Contained by:** module galacticus_nodes

**subroutine:** node_component_massflowstatistics_dump
*Description:* Dump the contents of a generic massFlowStatistics component.
*Code lines:* 10
**Contained by:** module galacticus_nodes
**Modules used:** galacticus_display iso_varying_string
subroutine: node_component_massflowstatistics_initializor
Description: Initialize a generic massFlowStatistics component.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_massflowstatistics_move
Description: Move instances of the massFlowStatistics component, from one node to another.
Code lines: 67
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_massflowstatistics_output
Description: Output properties for a massFlowStatistics component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatistics_output_count
Description: Increment the count of properties to output for a generic massFlowStatistics component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatistics_output_names
Description: Establish property names for a generic massFlowStatistics component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatistics_remove
Description: Removes an instance of the massFlowStatistics component, shifting other instances to keep the array contiguous.
Code lines: 39
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: node_component_massflowstatistics_type
Description: Returns the type for the massFlowStatistics component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsnull_builder
Description: Build a null implementation of the massFlowStatistics component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_massflowstatisticsnull_count
Description: Return a count of the serialization of a null implementation of the massFlowStatistics component.
19.1. Program units

subroutine: node_component_massflowstatisticsnull_deserialize_rates
Description: Serialize rates of a null implementation of the massFlowStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsnull_deserialize_scales
Description: Serialize scales of a null implementation of the massFlowStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsnull_deserialize_values
Description: Serialize values of a null implementation of the massFlowStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsnull_destroy
Description: Destroy a null implementation of the massFlowStatistics component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_massflowstatisticsnull_dump
Description: Dump the contents of a null implementation of the massFlowStatistics component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: galacticus_display, iso_varying_string, string_handling

subroutine: node_component_massflowstatisticsnull_dump_raw
Description: Dump the contents of a null implementation of the massFlowStatistics component in binary.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsnull_dump_xml
Description: Dump the contents of a null implementation of the massFlowStatistics component to XML.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsnull_initializor
Description: Initialize a null implementation of the massFlowStatistics component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_massflowstatisticsnull_is_active
Description: Return true if the null implementation of the massFlowStatistics component is the active choice.
19. Source Code Documentation

subroutine: node_component_massflowstatisticsnull_name_from_index
Description: Return the name of the property of given index for a null implementation of the massFlowStatistics component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_massflowstatisticsnull_ode_step_rates_init
Description: Initialize rates in a null implementation of the massFlowStatistics component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsnull_odestepscalesinit
Description: Initialize scales in a null implementation of the massFlowStatistics component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsnull_output_count
Description: Increment output property count for a null implementation of the massFlowStatistics component.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsnull_output_names
Description: Establish property names for a null implementation of the massFlowStatistics component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsnull_read_raw
Description: Read the contents of a null implementation of the massFlowStatistics component in binary.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_massflowstatisticsnull_serialize_rates
Description: Serialize rates of a null implementation of the massFlowStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsnull_serialize_scales
Description: Serialize scales of a null implementation of the massFlowStatistics component.
Code lines: 7
Contained by: module galacticus_nodes
19.1. Program units

**subroutine:** node_component_massflowstatisticsnull_serialize_values

*Description:* Serialize values of a null implementation of the massFlowStatistics component.

*Code lines:* 7

*Contained by:* module galacticus_nodes

**function:** node_component_massflowstatisticsnull_type

*Description:* Returns the type for the null implementation of the massFlowStatistics component.

*Code lines:* 8

*Contained by:* module galacticus_nodes

**subroutine:** node_component_massflowstatisticsstandard_builder

*Description:* Build a standard implementation of the massFlowStatistics component.

*Code lines:* 19

*Modules used:* fox_dom galacticus_error memory_management

**function:** node_component_massflowstatisticsstandard_count

*Description:* Return a count of the serialization of a standard implementation of the massFlowStatistics component.

*Code lines:* 8

*Contained by:* module galacticus_nodes

**subroutine:** node_component_massflowstatisticsstandard_deserialize_rates

*Description:* Serialize rates of a standard implementation of the massFlowStatistics component.

*Code lines:* 11

*Contained by:* module galacticus_nodes

**subroutine:** node_component_massflowstatisticsstandard_deserialize_scales

*Description:* Serialize scales of a standard implementation of the massFlowStatistics component.

*Code lines:* 11

*Contained by:* module galacticus_nodes

**subroutine:** node_component_massflowstatisticsstandard_deserialize_values

*Description:* Serialize values of a standard implementation of the massFlowStatistics component.

*Code lines:* 11

*Contained by:* module galacticus_nodes

**subroutine:** node_component_massflowstatisticsstandard_destroy

*Description:* Destroy a standard implementation of the massFlowStatistics component.

*Code lines:* 7

*Modules used:* memory_management

**subroutine:** node_component_massflowstatisticsstandard_dump

*Description:* Dump the contents of a standard implementation of the massFlowStatistics component.

*Code lines:* 16

*Contained by:* module galacticus_nodes
subroutine: node_component_massflowstatisticsstandard_dump_raw
Description: Dump the contents of a standard implementation of the massFlowStatistics component in binary.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsstandard_dump_xml
Description: Dump the contents of a standard implementation of the massFlowStatistics component to XML.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsstandard_initializor
Description: Initialize a standard implementation of the massFlowStatistics component.
Code lines: 8
Contained by: module galacticus_nodes

function: node_component_massflowstatisticsstandard_is_active
Description: Return true if the standard implementation of the massFlowStatistics component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsstandard_name_from_index
Description: Return the name of the property of given index for a standard implementation of the massFlowStatistics component.
Code lines: 14
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsstandard_ode_step_rates_init
Description: Initialize rates in a standard implementation of the massFlowStatistics component for an ODE solver step.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsstandard_odestepscalesinit
Description: Initialize scales in a standard implementation of the massFlowStatistics component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsstandard_output_count
Description: Increment output property count for a standard implementation of the massFlowStatistics component.
Code lines: 10
Contained by: module galacticus_nodes
subroutine: node_component_massflowstatisticsstandard_output_names
Description: Establish property names for a standard implementation of the massFlowStatistics component.
Code lines: 24
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsstandard_read_raw
Description: Read the contents of a standard implementation of the massFlowStatistics component in binary.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_massflowstatisticsstandard_serialize_rates
Description: Serialize rates of a standard implementation of the massFlowStatistics component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsstandard_serialize_scales
Description: Serialize scales of a standard implementation of the massFlowStatistics component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_massflowstatisticsstandard_serialize_values
Description: Serialize values of a standard implementation of the massFlowStatistics component.
Code lines: 11
Contained by: module galacticus_nodes

function: node_component_massflowstatisticsstandard_type
Description: Returns the type for the standard implementation of the massFlowStatistics component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatistics_builder
Description: Build a generic mergingStatistics component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error

subroutine: node_component_mergingstatistics_destroy
Description: Destroys a mergingStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatistics_dump
Description: Dump the contents of a generic mergingStatistics component.
Code lines: 10
Contained by: module galacticus_nodes
subroutine: node_component_mergingstatistics_initializor
Description: Initialize a generic mergingStatistics component.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_mergingstatistics_move
Description: Move instances of the mergingStatistics component, from one node to another.
Code lines: 81
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_mergingstatistics_output
Description: Output properties for a mergingStatistics component.
Code lines: 21
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatistics_output_count
Description: Increment the count of properties to output for a generic mergingStatistics component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatistics_output_names
Description: Establish property names for a generic mergingStatistics component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatistics_remove
Description: Removes an instance of the mergingStatistics component, shifting other instances to keep the array contiguous.
Code lines: 47
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: node_component_mergingstatistics_type
Description: Returns the type for the mergingStatistics component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsnull_builder
Description: Build a null implementation of the mergingStatistics component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom memory_management galacticus_error
function: node_component_mergingstatisticsnull_count
   Description: Return a count of the serialization of a null implementation of the mergingStatistics component.
   Code lines: 7
   Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsnull_deserialize_rates
   Description: Serialize rates of a null implementation of the mergingStatistics component.
   Code lines: 7
   Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsnull_deserialize_scales
   Description: Serialize scales of a null implementation of the mergingStatistics component.
   Code lines: 7
   Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsnull_deserialize_values
   Description: Serialize values of a null implementation of the mergingStatistics component.
   Code lines: 7
   Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsnull_destroy
   Description: Destroy a null implementation of the mergingStatistics component.
   Code lines: 7
   Contained by: module galacticus_nodes
   Modules used: memory_management

subroutine: node_component_mergingstatisticsnull_dump
   Description: Dump the contents of a null implementation of the mergingStatistics component.
   Code lines: 9
   Contained by: module galacticus_nodes
   Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_mergingstatisticsnull_dump_raw
   Description: Dump the contents of a null implementation of the mergingStatistics component in binary.
   Code lines: 7
   Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsnull_dump_xml
   Description: Dump the contents of a null implementation of the mergingStatistics component to XML.
   Code lines: 7
   Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsnull_initializor
   Description: Initialize a null implementation of the mergingStatistics component.
   Code lines: 7
   Contained by: module galacticus_nodes
19. Source Code Documentation

Modules used: memory_management

**function:** node_component_mergingstatisticsnull_is_active
**Description:** Return true if the null implementation of the mergingStatistics component is the active choice.
**Code lines:** 6
**Contained by:** module galacticus_nodes

**subroutine:** node_component_mergingstatisticsnull_name_from_index
**Description:** Return the name of the property of given index for a null implementation of the mergingStatistics component.
**Code lines:** 9
**Contained by:** module galacticus_nodes
**Modules used:** iso_varying_string

**subroutine:** node_component_mergingstatisticsnull_ode_step_rates_init
**Description:** Initialize rates in a null implementation of the mergingStatistics component for an ODE solver step.
**Code lines:** 6
**Contained by:** module galacticus_nodes

**subroutine:** node_component_mergingstatisticsnull_odestepscalesinit
**Description:** Initialize scales in a null implementation of the mergingStatistics component for an ODE solver step.
**Code lines:** 5
**Contained by:** module galacticus_nodes

**subroutine:** node_component_mergingstatisticsnull_output_count
**Description:** Increment output property count for a null implementation of the mergingStatistics component.
**Code lines:** 9
**Contained by:** module galacticus_nodes

**subroutine:** node_component_mergingstatisticsnull_output_names
**Description:** Establish property names for a null implementation of the mergingStatistics component.
**Code lines:** 11
**Contained by:** module galacticus_nodes

**subroutine:** node_component_mergingstatisticsnull_read_raw
**Description:** Read the contents of a null implementation of the mergingStatistics component in binary.
**Code lines:** 8
**Contained by:** module galacticus_nodes
**Modules used:** memory_management

**subroutine:** node_component_mergingstatisticsnull_serialize_rates
**Description:** Serialize rates of a null implementation of the mergingStatistics component.
**Code lines:** 7
**Contained by:** module galacticus_nodes
subroutine: node_component_mergingstatisticsnull_serialize_scales
Description: Serialize scales of a null implementation of the mergingStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsnull_serialize_values
Description: Serialize values of a null implementation of the mergingStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_mergingstatisticsnull_type
Description: Returns the type for the null implementation of the mergingStatistics component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsrecent_builder
Description: Build a recent implementation of the mergingStatistics component.
Code lines: 22
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_mergingstatisticsrecent_count
Description: Return a count of the serialization of a recent implementation of the mergingStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsrecent_deserialize_rates
Description: Serialize rates of a recent implementation of the mergingStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsrecent_deserialize_scales
Description: Serialize scales of a recent implementation of the mergingStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsrecent_deserialize_values
Description: Serialize values of a recent implementation of the mergingStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsrecent_destroy
Description: Destroy a recent implementation of the mergingStatistics component.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: memory_management
subroutine: node_component_mergingstatisticsrecent_dump
Description: Dump the contents of a recent implementation of the mergingStatistics component.
Code lines: 21
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_mergingstatisticsrecent_dump_raw
Description: Dump the contents of a recent implementation of the mergingStatistics component in binary.
Code lines: 12
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsrecent_dump_xml
Description: Dump the contents of a recent implementation of the mergingStatistics component to XML.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsrecent_initializor
Description: Initialize a recent implementation of the mergingStatistics component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: galacticus_output_times memory_management

function: node_component_mergingstatisticsrecent_is_active
Description: Return true if the recent implementation of the mergingStatistics component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsrecent_name_from_index
Description: Return the name of the property of given index for a recent implementation of the mergingStatistics component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_mergingstatisticsrecent_ode_step_rates_init
Description: Initialize rates in a recent implementation of the mergingStatistics component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsrecent_odestepsscalesinit
Description: Initialize scales in a recent implementation of the mergingStatistics component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsrecent_output_count
19.1. Program units

*Description:* Increment output property count for a recent implementation of the mergingStatistics component.
*Code lines:* 9
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_mergingstatisticsrecent_output_names`
*Description:* Establish property names for a recent implementation of the mergingStatistics component.
*Code lines:* 11
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_mergingstatisticsrecent_read_raw`
*Description:* Read the contents of a recent implementation of the mergingStatistics component in binary.
*Code lines:* 16
*Contained by:* module `galacticus_nodes`
*Modules used:* `memory_management`

**subroutine:** `node_component_mergingstatisticsrecent_serialize_rates`
*Description:* Serialize rates of a recent implementation of the mergingStatistics component.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_mergingstatisticsrecent_serialize_scales`
*Description:* Serialize scales of a recent implementation of the mergingStatistics component.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_mergingstatisticsrecent_serialize_values`
*Description:* Serialize values of a recent implementation of the mergingStatistics component.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**function:** `node_component_mergingstatisticsrecent_type`
*Description:* Returns the type for the recent implementation of the mergingStatistics component.
*Code lines:* 8
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_mergingstatisticsstandard_builder`
*Description:* Build a standard implementation of the mergingStatistics component.
*Code lines:* 37
*Contained by:* module `galacticus_nodes`
*Modules used:* `fox_dom` `galacticus_error` `memory_management`

**function:** `node_component_mergingstatisticsstandard_count`
*Description:* Return a count of the serialization of a standard implementation of the mergingStatistics component.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_mergingstatisticsstandard_deserialize_rates`
19. Source Code Documentation

**Description:** Serialize rates of a standard implementation of the mergingStatistics component.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**subroutine:** node_component_mergingstatisticsstandard_deserialize_scales

**Description:** Serialize scales of a standard implementation of the mergingStatistics component.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**subroutine:** node_component_mergingstatisticsstandard_deserialize_values

**Description:** Serialize values of a standard implementation of the mergingStatistics component.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**subroutine:** node_component_mergingstatisticsstandard_destroy

**Description:** Destroy a standard implementation of the mergingStatistics component.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**Modules used:** memory_management

**subroutine:** node_component_mergingstatisticsstandard_dump

**Description:** Dump the contents of a standard implementation of the mergingStatistics component.

**Code lines:** 25

**Contained by:** module galacticus_nodes

**Modules used:** galacticus_display iso_varying_string string_handling

**subroutine:** node_component_mergingstatisticsstandard_dump_raw

**Description:** Dump the contents of a standard implementation of the mergingStatistics component in binary.

**Code lines:** 11

**Contained by:** module galacticus_nodes

**subroutine:** node_component_mergingstatisticsstandard_dump_xml

**Description:** Dump the contents of a standard implementation of the mergingStatistics component to XML.

**Code lines:** 13

**Contained by:** module galacticus_nodes

**subroutine:** node_component_mergingstatisticsstandard_initializor

**Description:** Initialize a standard implementation of the mergingStatistics component.

**Code lines:** 11

**Contained by:** module galacticus_nodes

**Modules used:** memory_management

**function:** node_component_mergingstatisticsstandard_is_active

**Description:** Return true if the standard implementation of the mergingStatistics component is the active choice.

**Code lines:** 6

**Contained by:** module galacticus_nodes
subroutine: node_component_mergingstatisticsstandard_name_from_index
Description: Return the name of the property of given index for a standard implementation of the mergingStatistics component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_mergingstatisticsstandard_ode_step_rates_init
Description: Initialize rates in a standard implementation of the mergingStatistics component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsstandard_ode_step_scalesinit
Description: Initialize scales in a standard implementation of the mergingStatistics component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsstandard_output_count
Description: Increment output property count for a standard implementation of the mergingStatistics component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsstandard_output_names
Description: Establish property names for a standard implementation of the mergingStatistics component.
Code lines: 63
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsstandard_read_raw
Description: Read the contents of a standard implementation of the mergingStatistics component in binary.
Code lines: 12
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_mergingstatisticsstandard_serialize_rates
Description: Serialize rates of a standard implementation of the mergingStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsstandard_serialize_scales
Description: Serialize scales of a standard implementation of the mergingStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_mergingstatisticsstandard_serialize_values

Source Code Documentation

Description: Serialize values of a standard implementation of the mergingStatistics component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_mergingstatisticsstandard_type
Description: Returns the type for the standard implementation of the mergingStatistics component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_name_from_index
Description: Return the name of the property of given index.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_nbody_builder
Description: Build a generic nBody component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error

subroutine: node_component_nbody_destroy
Description: Destroys a nBody component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_nbody_dump
Description: Dump the contents of a generic nBody component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string

subroutine: node_component_nbody_initializor
Description: Initialize a generic nBody component.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_nbody_move
Description: Move instances of the nBody component, from one node to another.
Code lines: 67
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_nbody_output
Description: Output properties for a nBody component.
Code lines: 19
Contained by: module galacticus_nodes
subroutine: node_component_nbody_output_count
Description: Increment the count of properties to output for a generic nBody component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_nbody_output_names
Description: Establish property names for a generic nBody component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_nbody_remove
Description: Removes an instance of the nBody component, shifting other instances to keep the array contiguous.
Code lines: 39
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: node_component_nbody_type
Description: Returns the type for the nBody component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_nbodynull_builder
Description: Build a null implementation of the nBody component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_nbodynull_count
Description: Return a count of the serialization of a null implementation of the nBody component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_nbodynull_deserialize_rates
Description: Serialize rates of a null implementation of the nBody component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_nbodynull_deserialize_scales
Description: Serialize scales of a null implementation of the nBody component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_nbodynull_deserialize_values
Description: Serialize values of a null implementation of the nBody component.
Code lines: 7
Contained by: module galacticus_nodes
19. Source Code Documentation

**subroutine**: node_component_nbodynull_destroy

*Description:* Destroy a null implementation of the nBody component.

*Code lines:* 7

*Contained by:* module galacticus_nodes

*Modules used:* memory_management

**subroutine**: node_component_nbodynull_dump

*Description:* Dump the contents of a null implementation of the nBody component.

*Code lines:* 9

*Contained by:* module galacticus_nodes

*Modules used:* galacticus_display iso_varying_string string_handling

**subroutine**: node_component_nbodynull_dump_raw

*Description:* Dump the contents of a null implementation of the nBody component in binary.

*Code lines:* 7

*Contained by:* module galacticus_nodes

**subroutine**: node_component_nbodynull_dump_xml

*Description:* Dump the contents of a null implementation of the nBody component to XML.

*Code lines:* 7

*Contained by:* module galacticus_nodes

**subroutine**: node_component_nbodynull_initializor

*Description:* Initialize a null implementation of the nBody component.

*Code lines:* 7

*Contained by:* module galacticus_nodes

*Modules used:* memory_management

**function**: node_component_nbodynull_is_active

*Description:* Return true if the null implementation of the nBody component is the active choice.

*Code lines:* 6

*Contained by:* module galacticus_nodes

**subroutine**: node_component_nbodynull_name_from_index

*Description:* Return the name of the property of given index for a null implementation of the nBody component.

*Code lines:* 9

*Contained by:* module galacticus_nodes

*Modules used:* iso_varying_string

**subroutine**: node_component_nbodynull_ode_step_rates_init

*Description:* Initialize rates in a null implementation of the nBody component for an ODE solver step.

*Code lines:* 6

*Contained by:* module galacticus_nodes

**subroutine**: node_component_nbodynull_odestepscalesinit

*Description:* Initialize scales in a null implementation of the nBody component for an ODE solver step.
19.1. Program units

Code lines: 5
Contained by: module galacticus_nodes

**subroutine: node_component_nbodynull_output_count**
*Description:* Increment output property count for a null implementation of the nBody component.
*Code lines:* 9
*Contained by:* module galacticus_nodes

**subroutine: node_component_nbodynull_output_names**
*Description:* Establish property names for a null implementation of the nBody component.
*Code lines:* 11
*Contained by:* module galacticus_nodes

**subroutine: node_component_nbodynull_read_raw**
*Description:* Read the contents of a null implementation of the nBody component in binary.
*Code lines:* 8
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**subroutine: node_component_nbodynull_serialize_rates**
*Description:* Serialize rates of a null implementation of the nBody component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine: node_component_nbodynull_serialize_scales**
*Description:* Serialize scales of a null implementation of the nBody component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine: node_component_nbodynull_serialize_values**
*Description:* Serialize values of a null implementation of the nBody component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**function: node_component_nbodynull_type**
*Description:* Returns the type for the null implementation of the nBody component.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine: node_component_nbodystandard_builder**
*Description:* Build a standard implementation of the nBody component.
*Code lines:* 31
*Contained by:* module galacticus_nodes
*Modules used:* fox_dom memory_management galacticus_error

**function: node_component_nbodystandard_count**
*Description:* Return a count of the serialization of a standard implementation of the nBody component.
*Code lines:* 7
19. Source Code Documentation

**Contained by:** module galacticus_nodes

**subroutine:** node_component_nbodystandard_deserialize_rates
*Description:* Serialize rates of a standard implementation of the nBody component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** node_component_nbodystandard_deserialize_scales
*Description:* Serialize scales of a standard implementation of the nBody component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** node_component_nbodystandard_deserialize_values
*Description:* Serialize values of a standard implementation of the nBody component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** node_component_nbodystandard_destroy
*Description:* Destroy a standard implementation of the nBody component.
*Code lines:* 7
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**subroutine:** node_component_nbodystandard_dump
*Description:* Dump the contents of a standard implementation of the nBody component.
*Code lines:* 22
*Contained by:* module galacticus_nodes
*Modules used:* galacticus_display, iso_varying_string, string_handling

**subroutine:** node_component_nbodystandard_dump_raw
*Description:* Dump the contents of a standard implementation of the nBody component in binary.
*Code lines:* 10
*Contained by:* module galacticus_nodes

**subroutine:** node_component_nbodystandard_dump_xml
*Description:* Dump the contents of a standard implementation of the nBody component to XML.
*Code lines:* 12
*Contained by:* module galacticus_nodes

**subroutine:** node_component_nbodystandard_initializor
*Description:* Initialize a standard implementation of the nBody component.
*Code lines:* 10
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**function:** node_component_nbodystandard_is_active
*Description:* Return true if the standard implementation of the nBody component is the active choice.
*Code lines:* 6
19.1. Program units

**module** galacticus_nodes

**subroutine** node_component_nbodystandard_name_from_index

*Description:* Return the name of the property of given index for a standard implementation of the nBody component.

*Code lines:* 9

**Contained by:** module galacticus_nodes

**Modules used:** iso_varying_string

**subroutine** node_component_nbodystandard_ode_step_rates_init

*Description:* Initialize rates in a standard implementation of the nBody component for an ODE solver step.

*Code lines:* 6

**Contained by:** module galacticus_nodes

**subroutine** node_component_nbodystandard_odestepscalesinit

*Description:* Initialize scales in a standard implementation of the nBody component for an ODE solver step.

*Code lines:* 5

**Contained by:** module galacticus_nodes

**subroutine** node_component_nbodystandard_output_count

*Description:* Increment output property count for a standard implementation of the nBody component.

*Code lines:* 11

**Contained by:** module galacticus_nodes

**subroutine** node_component_nbodystandard_output_names

*Description:* Establish property names for a standard implementation of the nBody component.

*Code lines:* 50

**Contained by:** module galacticus_nodes

**subroutine** node_component_nbodystandard_read_raw

*Description:* Read the contents of a standard implementation of the nBody component in binary.

*Code lines:* 11

**Contained by:** module galacticus_nodes

**Modules used:** memory_management

**subroutine** node_component_nbodystandard_serialize_rates

*Description:* Serialize rates of a standard implementation of the nBody component.

*Code lines:* 7

**Contained by:** module galacticus_nodes

**subroutine** node_component_nbodystandard_serialize_scales

*Description:* Serialize scales of a standard implementation of the nBody component.

*Code lines:* 7

**Contained by:** module galacticus_nodes

**subroutine** node_component_nbodystandard_serialize_values

*Description:* Serialize values of a standard implementation of the nBody component.
19. Source Code Documentation

Code lines: 7
Contained by: module galacticus_nodes

function: node_component_nbodystandard_type
Description: Returns the type for the standard implementation of the nBody component.
Code lines: 8
Contained by: module galacticus_nodes

function: node_component_null_double0_inout
Description: A null double function for rank 0 nodeComponent arrays.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_null_void0_inout
Description: A null void function for rank 0 nodeComponent arrays.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_ode_step_initialize_null
Description: Initialize a generic tree node component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_output_count_null
Description: Dump a generic tree node component.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_output_names_null
Description: Dump a generic tree node component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_output_null
Description: Dump a generic tree node component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_position_builder
Description: Build a generic position component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error

subroutine: node_component_position_destroy
Description: Destroys a position component.
Code lines: 7
Contained by: module galacticus_nodes
### 19.1. Program units

**subroutine:** node_component_position_dump  
*Description:* Dump the contents of a generic position component.  
*Code lines:* 10  
*Contained by:* module galacticus_nodes  
*Modules used:* galacticus_display, iso_varying_string

**subroutine:** node_component_position_initializor  
*Description:* Initialize a generic position component.  
*Code lines:* 8  
*Contained by:* module galacticus_nodes  
*Modules used:* galacticus_error

**subroutine:** node_component_position_move  
*Description:* Move instances of the position component, from one node to another.  
*Code lines:* 67  
*Contained by:* module galacticus_nodes  
*Modules used:* galacticus_error

**subroutine:** node_component_position_output  
*Description:* Output properties for a position component.  
*Code lines:* 17  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_position_output_count  
*Description:* Increment the count of properties to output for a generic position component.  
*Code lines:* 13  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_position_output_names  
*Description:* Establish property names for a generic position component.  
*Code lines:* 15  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_position_remove  
*Description:* Removes an instance of the position component, shifting other instances to keep the array contiguous.  
*Code lines:* 39  
*Contained by:* module galacticus_nodes  
*Modules used:* galacticus_error

**function:** node_component_position_type  
*Description:* Returns the type for the position component.  
*Code lines:* 8  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_positionnull_builder  
*Description:* Build a null implementation of the position component.  
*Code lines:* 10
19. Source Code Documentation

function: node_component_positionnull_count
Description: Return a count of the serialization of a null implementation of the position component.
Code lines: 7
Contained by: module galacticus_nodes

module used: fox_dom galacticus_error memory_management

subroutine: node_component_positionnull_deserialize_rates
Description: Serialize rates of a null implementation of the position component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_positionnull_deserialize_scales
Description: Serialize scales of a null implementation of the position component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_positionnull_deserialize_values
Description: Serialize values of a null implementation of the position component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_positionnull_destroy
Description: Destroy a null implementation of the position component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_positionnull_dump
Description: Dump the contents of a null implementation of the position component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_positionnull_dump_raw
Description: Dump the contents of a null implementation of the position component in binary.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_positionnull_dump_xml
Description: Dump the contents of a null implementation of the position component to XML.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_positionnull_initializor
Description: Initialize a null implementation of the position component.
19.1. Program units

**Code lines:** 7  
**Contained by:** module galacticus_nodes  
**Modules used:** memory_management

**function:** node_component_positionnull_is_active  
**Description:** Return true if the null implementation of the position component is the active choice.  
**Code lines:** 6  
**Contained by:** module galacticus_nodes

**subroutine:** node_component_positionnull_name_from_index  
**Description:** Return the name of the property of given index for a null implementation of the position component.  
**Code lines:** 9  
**Contained by:** module galacticus_nodes  
**Modules used:** iso_varying_string

**subroutine:** node_component_positionnull_ode_step_rates_init  
**Description:** Initialize rates in a null implementation of the position component for an ODE solver step.  
**Code lines:** 6  
**Contained by:** module galacticus_nodes

**subroutine:** node_component_positionnull_ode_stepsccalesinit  
**Description:** Initialize scales in a null implementation of the position component for an ODE solver step.  
**Code lines:** 5  
**Contained by:** module galacticus_nodes

**subroutine:** node_component_positionnull_output_count  
**Description:** Increment output property count for a null implementation of the position component.  
**Code lines:** 9  
**Contained by:** module galacticus_nodes

**subroutine:** node_component_positionnull_output_names  
**Description:** Establish property names for a null implementation of the position component.  
**Code lines:** 11  
**Contained by:** module galacticus_nodes

**subroutine:** node_component_positionnull_read_raw  
**Description:** Read the contents of a null implementation of the position component in binary.  
**Code lines:** 8  
**Contained by:** module galacticus_nodes  
**Modules used:** memory_management

**subroutine:** node_component_positionnull_serialize_rates  
**Description:** Serialize rates of a null implementation of the position component.  
**Code lines:** 7  
**Contained by:** module galacticus_nodes

**subroutine:** node_component_positionnull_serialize_scales  
**Description:** Serialize scales of a null implementation of the position component.
19. Source Code Documentation

subroutine: node_component_positionnull_serialize_values
Description: Serialize values of a null implementation of the position component.
Code lines: 7
Contained by: module galacticus_nodes

function: node_component_positionnull_type
Description: Returns the type for the null implementation of the position component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_positionpreset_builder
Description: Build a preset implementation of the position component.
Code lines: 36
Contained by: module galacticus_nodes
Modules used: fox_dom memory_management galacticus_error

function: node_component_positionpreset_count
Description: Return a count of the serialization of a preset implementation of the position component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_positionpreset_deserialize_rates
Description: Serialize rates of a preset implementation of the position component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_positionpreset_deserialize_scales
Description: Serialize scales of a preset implementation of the position component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_positionpreset_deserialize_values
Description: Serialize values of a preset implementation of the position component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_positionpreset_destroy
Description: Destroy a preset implementation of the position component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_positionpreset_dump
Description: Dump the contents of a preset implementation of the position component.
Code lines: 32
19.1. Program units

Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_positionpreset_dump_raw
Description: Dump the contents of a preset implementation of the position component in binary.
Code lines: 18
Contained by: module galacticus_nodes

subroutine: node_component_positionpreset_dump_xml
Description: Dump the contents of a preset implementation of the position component to XML.
Code lines: 18
Contained by: module galacticus_nodes

subroutine: node_component_positionpreset_initializor
Description: Initialize a preset implementation of the position component.
Code lines: 12
Contained by: module galacticus_nodes

Modules used: memory_management

function: node_component_positionpreset_is_active
Description: Return true if the preset implementation of the position component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_positionpreset_name_from_index
Description: Return the name of the property of given index for a preset implementation of the position component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_positionpreset_ode_step_rates_init
Description: Initialize rates in a preset implementation of the position component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_positionpreset_odestepscalesinit
Description: Initialize scales in a preset implementation of the position component for an ODE solver step.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: node_component_positionpreset_output_count
Description: Increment output property count for a preset implementation of the position component.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_positionpreset_output_names
1583
19. Source Code Documentation

**Description:** Establish property names for a preset implementation of the position component.

**Code lines:** 53

**Contained by:** module galacticus_nodes

**subroutine: node_component_positionpreset_read_raw**

**Description:** Read the contents of a preset implementation of the position component in binary.

**Code lines:** 23

**Contained by:** module galacticus_nodes

**Modules used:** memory_management

**subroutine: node_component_positionpreset_serialize_rates**

**Description:** Serialize rates of a preset implementation of the position component.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**subroutine: node_component_positionpreset_serialize_scales**

**Description:** Serialize scales of a preset implementation of the position component.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**subroutine: node_component_positionpreset_serialize_values**

**Description:** Serialize values of a preset implementation of the position component.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**function: node_component_positionpreset_type**

**Description:** Returns the type for the preset implementation of the position component.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**function: node_component_potential_null**

**Description:** A null implementation of the gravitational potential in a component. Always returns zero.

**Code lines:** 10

**Contained by:** module galacticus_nodes

**function: node_component_rotation_curve_gradient_null**

**Description:** A null implementation of the gradient of the rotation curve due to a component. Always returns zero.

**Code lines:** 10

**Contained by:** module galacticus_nodes

**function: node_component_rotation_curve_null**

**Description:** A null implementation of the rotation curve due to a component. Always returns zero.

**Code lines:** 10

**Contained by:** module galacticus_nodes

**subroutine: node_component_satellite_builder**

**Description:** Build a generic satellite component.

**Code lines:** 10
19.1. Program units

Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error

subroutine: node_component_satellite_destroy
Description: Destroys a satellite component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_satellite_dump
Description: Dump the contents of a generic satellite component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string

subroutine: node_component_satellite_initializor
Description: Initialize a generic satellite component.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_satellite_move
Description: Move instances of the satellite component, from one node to another.
Code lines: 109
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: node_component_satellite_orbiting_time_of_merging
Description: Return the timeOfMerging property of the satelliteOrbiting component class.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: node_component_satellite_output
Description: Output properties for a satellite component.
Code lines: 44
Contained by: module galacticus_nodes

subroutine: node_component_satellite_output_count
Description: Increment the count of properties to output for a generic satellite component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_satellite_output_names
Description: Establish property names for a generic satellite component.
Code lines: 15
Contained by: module galacticus_nodes

function: node_component_satellite_preset_merge_time
Description: Return the mergeTime property of the satellitePreset component class.
Code lines: 15
19. Source Code Documentation

```
module galacticus_nodes

subroutine: node_component_satellite_preset_merge_time_set
  
  Description: Sets the mergeTime property of the satellitePreset component class.
  
  Code lines: 12
  Contained by: module galacticus_nodes

subroutine: node_component_satellite_remove
  
  Description: Removes an instance of the satellite component, shifting other instances to keep the array contiguous.
  
  Code lines: 63
  Contained by: module galacticus_nodes
  Modules used: galacticus_error

function: node_component_satellite_standard_merge_time
  
  Description: Return the mergeTime property of the satelliteStandard component.
  
  Code lines: 13
  Contained by: module galacticus_nodes

function: node_component_satellite_standard_time_of_merging
  
  Description: Return the timeOfMerging property of the satelliteStandard component class.
  
  Code lines: 15
  Contained by: module galacticus_nodes

function: node_component_satellite_type
  
  Description: Returns the type for the satellite component.
  
  Code lines: 8
  Contained by: module galacticus_nodes

function: node_component_satellite_very_simple_merge_time
  
  Description: Return the mergeTime property of the satelliteVerySimple component.
  
  Code lines: 13
  Contained by: module galacticus_nodes

function: node_component_satellite_very_simple_time_of_merging
  
  Description: Return the timeOfMerging property of the satelliteVerySimple component.
  
  Code lines: 16
  Contained by: module galacticus_nodes

subroutine: node_component_satellitenull_builder
  
  Description: Build a null implementation of the satellite component.
  
  Code lines: 10
  Contained by: module galacticus_nodes
  Modules used: fox_dom
               galacticus_error
               memory_management

function: node_component_satellitenull_count
  
  Description: Return a count of the serialization of a null implementation of the satellite component.
  
  Code lines: 7
```

19.1. Program units

**Contained by:** module galacticus_nodes

**subroutine:** node_component_satellitenull_deserialize_rates
*Description:* Serialize rates of a null implementation of the satellite component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** node_component_satellitenull_deserialize_scales
*Description:* Serialize scales of a null implementation of the satellite component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** node_component_satellitenull_deserialize_values
*Description:* Serialize values of a null implementation of the satellite component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** node_component_satellitenull_destroy
*Description:* Destroy a null implementation of the satellite component.
*Code lines:* 7
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**subroutine:** node_component_satellitenull_dump
*Description:* Dump the contents of a null implementation of the satellite component.
*Code lines:* 9
*Contained by:* module galacticus_nodes
*Modules used:* galacticus_display iso_varying_string string_handling

**subroutine:** node_component_satellitenull_dump_raw
*Description:* Dump the contents of a null implementation of the satellite component in binary.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** node_component_satellitenull_dump_xml
*Description:* Dump the contents of a null implementation of the satellite component to XML.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** node_component_satellitenull_initializor
*Description:* Initialize a null implementation of the satellite component.
*Code lines:* 7
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**function:** node_component_satellitenull_is_active
*Description:* Return true if the null implementation of the satellite component is the active choice.
*Code lines:* 6
19. Source Code Documentation

**Contained by:** module galacticus_nodes

**subroutine:** node_component_satellitenull_name_from_index

*Description:* Return the name of the property of given index for a null implementation of the satellite component.

*Code lines:* 9

**Contained by:** module galacticus_nodes

**Modules used:** iso_varying_string

**subroutine:** node_component_satellitenull_ode_step_rates_init

*Description:* Initialize rates in a null implementation of the satellite component for an ODE solver step.

*Code lines:* 6

**Contained by:** module galacticus_nodes

**subroutine:** node_component_satellitenull_odestepscalesinit

*Description:* Initialize scales in a null implementation of the satellite component for an ODE solver step.

*Code lines:* 5

**Contained by:** module galacticus_nodes

**subroutine:** node_component_satellitenull_output_count

*Description:* Increment output property count for a null implementation of the satellite component.

*Code lines:* 9

**Contained by:** module galacticus_nodes

**subroutine:** node_component_satellitenull_output_names

*Description:* Establish property names for a null implementation of the satellite component.

*Code lines:* 11

**Contained by:** module galacticus_nodes

**subroutine:** node_component_satellitenull_read_raw

*Description:* Read the contents of a null implementation of the satellite component in binary.

*Code lines:* 8

**Modules used:** module galacticus_nodes

**subroutine:** node_component_satellitenull_serialize_rates

*Description:* Serialize rates of a null implementation of the satellite component.

*Code lines:* 7

**Contained by:** module galacticus_nodes

**subroutine:** node_component_satellitenull_serialize_scales

*Description:* Serialize scales of a null implementation of the satellite component.

*Code lines:* 7

**Contained by:** module galacticus_nodes

**subroutine:** node_component_satellitenull_serialize_values

*Description:* Serialize values of a null implementation of the satellite component.

*Code lines:* 7

**Contained by:** module galacticus_nodes
function: node_component_satellitenull_type
Description: Returns the type for the null implementation of the satellite component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_satelliteorbiting_builder
Description: Build a orbiting implementation of the satellite component.
Code lines: 64
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_satelliteorbiting_count
Description: Return a count of the serialization of a orbiting implementation of the satellite component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_satelliteorbiting_deserialize_rates
Description: Serialize rates of a orbiting implementation of the satellite component.
Code lines: 26
Contained by: module galacticus_nodes

subroutine: node_component_satelliteorbiting_deserialize_scales
Description: Serialize scales of a orbiting implementation of the satellite component.
Code lines: 26
Contained by: module galacticus_nodes

subroutine: node_component_satelliteorbiting_deserialize_values
Description: Serialize values of a orbiting implementation of the satellite component.
Code lines: 26
Contained by: module galacticus_nodes

subroutine: node_component_satelliteorbiting_destroy
Description: Destroy a orbiting implementation of the satellite component.
Code lines: 17
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_satelliteorbiting_dump
Description: Dump the contents of a orbiting implementation of the satellite component.
Code lines: 45
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_satelliteorbiting_dump_raw
Description: Dump the contents of a orbiting implementation of the satellite component in binary.
Code lines: 22
Contained by: module galacticus_nodes

**subroutine**: node_component_satelliteorbiting_dump_xml
*Description:* Dump the contents of an orbiting implementation of the satellite component to XML.
*Code lines:* 24
*Contained by:* module galacticus_nodes

**subroutine**: node_component_satelliteorbiting_initializor
*Description:* Initialize an orbiting implementation of the satellite component.
*Code lines:* 22
*Contained by:* module galacticus_nodes

**Modules used:**
- memory_management

**function**: node_component_satelliteorbiting_is_active
*Description:* Return true if the orbiting implementation of the satellite component is the active choice.
*Code lines:* 6
*Contained by:* module galacticus_nodes

**subroutine**: node_component_satelliteorbiting_name_from_index
*Description:* Return the name of the property of given index for an orbiting implementation of the satellite component.
*Code lines:* 34
*Contained by:* module galacticus_nodes

**Modules used:**
- iso_varying_string

**subroutine**: node_component_satelliteorbiting_ode_step_rates_init
*Description:* Initialize rates in an orbiting implementation of the satellite component for an ODE solver step.
*Code lines:* 11
*Contained by:* module galacticus_nodes

**subroutine**: node_component_satelliteorbiting_odestepscalesinit
*Description:* Initialize scales in an orbiting implementation of the satellite component for an ODE solver step.
*Code lines:* 10
*Contained by:* module galacticus_nodes

**subroutine**: node_component_satelliteorbiting_output_count
*Description:* Increment output property count for an orbiting implementation of the satellite component.
*Code lines:* 10
*Contained by:* module galacticus_nodes

**subroutine**: node_component_satelliteorbiting_output_names
*Description:* Establish property names for an orbiting implementation of the satellite component.
*Code lines:* 79
*Contained by:* module galacticus_nodes

**subroutine**: node_component_satelliteorbiting_read_raw
*Description:* Read the contents of an orbiting implementation of the satellite component in binary.
19.1. Program units

Code lines: 31
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_satelliteorbiting_serialize_rates
Description: Serialize rates of a orbiting implementation of the satellite component.
Code lines: 26
Contained by: module galacticus_nodes

subroutine: node_component_satelliteorbiting_serialize_scales
Description: Serialize scales of a orbiting implementation of the satellite component.
Code lines: 26
Contained by: module galacticus_nodes

subroutine: node_component_satelliteorbiting_serialize_values
Description: Serialize values of a orbiting implementation of the satellite component.
Code lines: 26
Contained by: module galacticus_nodes

function: node_component_satelliteorbiting_type
Description: Returns the type for the orbiting implementation of the satellite component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_satellitepreset_builder
Description: Build a preset implementation of the satellite component.
Code lines: 43
Contained by: module galacticus_nodes
Modules used: fox_dom memory_management galacticus_error

function: node_component_satellitepreset_count
Description: Return a count of the serialization of a preset implementation of the satellite component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_satellitepreset_deserialize_rates
Description: Serialize rates of a preset implementation of the satellite component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_satellitepreset_deserialize_scales
Description: Serialize scales of a preset implementation of the satellite component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_satellitepreset_deserialize_values
Description: Serialize values of a preset implementation of the satellite component.
Code lines: 7
subroutine: node_component_satellitepreset_destroy
Description: Destroy a preset implementation of the satellite component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_satellitepreset_dump
Description: Dump the contents of a preset implementation of the satellite component.
Code lines: 31
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_satellitepreset_dump_raw
Description: Dump the contents of a preset implementation of the satellite component in binary.
Code lines: 12
Contained by: module galacticus_nodes

subroutine: node_component_satellitepreset_dump_xml
Description: Dump the contents of a preset implementation of the satellite component to XML.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: node_component_satellitepreset_initializor
Description: Initialize a preset implementation of the satellite component.
Code lines: 12
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_satellitepreset_is_active
Description: Return true if the preset implementation of the satellite component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_satellitepreset_name_from_index
Description: Return the name of the property of given index for a preset implementation of the satellite component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_satellitepreset_ode_step_rates_init
Description: Initialize rates in a preset implementation of the satellite component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_satellitepreset_odestepscalesinit
**Description**: Initialize scales in a preset implementation of the satellite component for an ODE solver step.

**Code lines**: 5

**Contained by**: module `galacticus_nodes`

**subroutine**: `node_component_satellitepreset_output_count`

**Description**: Increment output property count for a preset implementation of the satellite component.

**Code lines**: 11

**Contained by**: module `galacticus_nodes`

**subroutine**: `node_component_satellitepreset_output_names`

**Description**: Establish property names for a preset implementation of the satellite component.

**Code lines**: 50

**Contained by**: module `galacticus_nodes`

**subroutine**: `node_component_satellitepreset_read_raw`

**Description**: Read the contents of a preset implementation of the satellite component in binary.

**Code lines**: 13

**Contained by**: module `galacticus_nodes`

**Modules used**: `memory_management`

**subroutine**: `node_component_satellitepreset_serialize_rates`

**Description**: Serialize rates of a preset implementation of the satellite component.

**Code lines**: 7

**Contained by**: module `galacticus_nodes`

**subroutine**: `node_component_satellitepreset_serialize_scales`

**Description**: Serialize scales of a preset implementation of the satellite component.

**Code lines**: 7

**Contained by**: module `galacticus_nodes`

**subroutine**: `node_component_satellitepreset_serialize_values`

**Description**: Serialize values of a preset implementation of the satellite component.

**Code lines**: 7

**Contained by**: module `galacticus_nodes`

**function**: `node_component_satellitepreset_type`

**Description**: Returns the type for the preset implementation of the satellite component.

**Code lines**: 8

**Contained by**: module `galacticus_nodes`

**subroutine**: `node_component_satellitestandard_builder`

**Description**: Build a standard implementation of the satellite component.

**Code lines**: 31

**Contained by**: module `galacticus_nodes`

**Modules used**: `fox_dom galacticus_error memory_management`
function: node_component_satellitestandard_count
Description: Return a count of the serialization of a standard implementation of the satellite component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_satellitestandard_deserialize_rates
Description: Serialize rates of a standard implementation of the satellite component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_satellitestandard_deserialize_scales
Description: Serialize scales of a standard implementation of the satellite component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_satellitestandard_deserialize_values
Description: Serialize values of a standard implementation of the satellite component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_satellitestandard_destroy
Description: Destroy a standard implementation of the satellite component.
Code lines: 8
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_satellitestandard_dump
Description: Dump the contents of a standard implementation of the satellite component.
Code lines: 23
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_satellitestandard_dump_raw
Description: Dump the contents of a standard implementation of the satellite component in binary.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_satellitestandard_dump_xml
Description: Dump the contents of a standard implementation of the satellite component to XML.
Code lines: 14
Contained by: module galacticus_nodes

subroutine: node_component_satellitestandard_initializor
Description: Initialize a standard implementation of the satellite component.
Code lines: 12
Contained by: module galacticus_nodes
19.1. Program units

Modules used: memory_management

function: node_component_satellitestandard_is_active
Description: Return true if the standard implementation of the satellite component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_satellitestandard_name_from_index
Description: Return the name of the property of given index for a standard implementation of the satellite component.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_satellitestandard_ode_step_rates_init
Description: Initialize rates in a standard implementation of the satellite component for an ODE solver step.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_satellitestandard_odestepscalesinit
Description: Initialize scales in a standard implementation of the satellite component for an ODE solver step.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_satellitestandard_output_count
Description: Increment output property count for a standard implementation of the satellite component.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_satellitestandard_output_names
Description: Establish property names for a standard implementation of the satellite component.
Code lines: 42
Contained by: module galacticus_nodes

subroutine: node_component_satellitestandard_read_raw
Description: Read the contents of a standard implementation of the satellite component in binary.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_satellitestandard_serialize_rates
Description: Serialize rates of a standard implementation of the satellite component.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_satellitestandard_serialize_scales
Description: Serialize scales of a standard implementation of the satellite component.
subroutine: node_component_satellitestandard_serialize_values
Description: Serialize values of a standard implementation of the satellite component.
Code lines: 13
Contained by: module galacticus_nodes

function: node_component_satellitestandard_type
Description: Returns the type for the standard implementation of the satellite component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_satelliteverysimple_builder
Description: Build a verySimple implementation of the satellite component.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_satelliteverysimple_count
Description: Return a count of the serialization of a verySimple implementation of the satellite component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_satelliteverysimple_deserialize_rates
Description: Serialize rates of a verySimple implementation of the satellite component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_satelliteverysimple_deserialize_scales
Description: Serialize scales of a verySimple implementation of the satellite component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_satelliteverysimple_deserialize_values
Description: Serialize values of a verySimple implementation of the satellite component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_satelliteverysimple_destroy
Description: Destroy a verySimple implementation of the satellite component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_satelliteverysimple_dump
Description: Dump the contents of a verySimple implementation of the satellite component.
19.1. Program units

subroutine: node_component_satelliteverysimple_dump_raw
Description: Dump the contents of a verySimple implementation of the satellite component in binary.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_satelliteverysimple_dump_xml
Description: Dump the contents of a verySimple implementation of the satellite component to XML.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_satelliteverysimple_initializor
Description: Initialize a verySimple implementation of the satellite component.
Code lines: 8
Contained by: module galacticus_nodes

function: node_component_satelliteverysimple_is_active
Description: Return true if the verySimple implementation of the satellite component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_satelliteverysimple_name_from_index
Description: Return the name of the property of given index for a verySimple implementation of the satellite component.
Code lines: 14
Contained by: module galacticus_nodes

subroutine: node_component_satelliteverysimple_ode_step_rates_init
Description: Initialize rates in a verySimple implementation of the satellite component for an ODE solver step.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_satelliteverysimple_odestepscalesinit
Description: Initialize scales in a verySimple implementation of the satellite component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_satelliteverysimple_output_count
Description: Increment output property count for a verySimple implementation of the satellite component.
Code lines: 10
Contained by: module galacticus_nodes
subroutine: node_component_satelliteverysimple_output_names
Description: Establish property names for a verySimple implementation of the satellite component.
Code lines: 24
Contained by: module galacticus_nodes

subroutine: node_component_satelliteverysimple_read_raw
Description: Read the contents of a verySimple implementation of the satellite component in binary.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_satelliteverysimple_serialize_rates
Description: Serialize rates of a verySimple implementation of the satellite component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_satelliteverysimple_serialize_scales
Description: Serialize scales of a verySimple implementation of the satellite component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_satelliteverysimple_serialize_values
Description: Serialize values of a verySimple implementation of the satellite component.
Code lines: 11
Contained by: module galacticus_nodes

function: node_component_satelliteverysimple_type
Description: Returns the type for the verySimple implementation of the satellite component.
Code lines: 8
Contained by: module galacticus_nodes

function: node_component_serialize_count_zero
Description: Return the serialization count of a generic tree node component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_serialize_null
Description: Serialize a generic tree node component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_spheroid_builder
Description: Build a generic spheroid component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error
subroutine: node_component_spheroid_destroy
Description:        Destroys a spheroid component.
Code lines:          7
Contained by:        module galacticus_nodes

subroutine: node_component_spheroid_dump
Description:        Dump the contents of a generic spheroid component.
Code lines:          10
Contained by:        module galacticus_nodes
Modules used:        galacticus_display iso_varying_string

subroutine: node_component_spheroid_initializor
Description:        Initialize a generic spheroid component.
Code lines:          8
Contained by:        module galacticus_nodes
Modules used:        galacticus_error

subroutine: node_component_spheroid_move
Description:        Move instances of the spheroid component, from one node to another.
Code lines:          67
Contained by:        module galacticus_nodes
Modules used:        galacticus_error

subroutine: node_component_spheroid_output
Description:        Output properties for a spheroid component.
Code lines:          35
Contained by:        module galacticus_nodes

subroutine: node_component_spheroid_output_count
Description:        Increment the count of properties to output for a generic spheroid component.
Code lines:          13
Contained by:        module galacticus_nodes

subroutine: node_component_spheroid_output_names
Description:        Establish property names for a generic spheroid component.
Code lines:          15
Contained by:        module galacticus_nodes

subroutine: node_component_spheroid_remove
Description:        Removes an instance of the spheroid component, shifting other instances to keep the array contiguous.
Code lines:          39
Contained by:        module galacticus_nodes
Modules used:        galacticus_error

function: node_component_spheroid_standard_density
Description:        Computes the density at a given position for an standard spheroid.
Code lines:          49
19. Source Code Documentation

Contains by: module galacticus_nodes
Modules used: coordinates galactic_structure_options
node_component_spheroid_standard_data numerical_constants_math

function node_component_spheroid_standard_enclosed_mass
Description: Computes the mass within a given radius for an standard spheroid.
Code lines: 45
Contained by: module galacticus_nodes
Modules used: galactic_structure_options node_component_spheroid_standard_data

function node_component_spheroid_standard_half_mass_radius
Description: Return the half-mass radius of the standard spheroid.
Code lines: 12
Contained by: module galacticus_nodes
Modules used: mass_distributions node_component_spheroid_standard_data

function node_component_spheroid_standard_potential
Description: Return the potential due to the standard spheroid.
Code lines: 30
Contained by: module galacticus_nodes
Modules used: coordinates galactic_structure_options
node_component_spheroid_standard_data numerical_constants_physical

function node_component_spheroid_standard_rotation_curve
Description: Computes the rotation curve at a given radius for a standard spheroid.
Code lines: 21
Contained by: module galacticus_nodes
Modules used: galactic_structure_options numerical_constants_physical

function node_component_spheroid_standard_rotation_curve_gradient
Description: Computes the rotation curve gradient for the standard spheroid.
Code lines: 26
Contained by: module galacticus_nodes
Modules used: galactic_structure_options numerical_constants_math
numerical_constants_physical numerical_constants_prefixes

function node_component_spheroid_type
Description: Returns the type for the spheroid component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine node_component_spheroidnull_builder
Description: Build a null implementation of the spheroid component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error
memory_management
function: node_component_spheroidnull_count
Description: Return a count of the serialization of a null implementation of the spheroid component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_spheroidnull_deserialize_rates
Description: Serialize rates of a null implementation of the spheroid component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_spheroidnull_deserialize_scales
Description: Serialize scales of a null implementation of the spheroid component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_spheroidnull_deserialize_values
Description: Serialize values of a null implementation of the spheroid component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_spheroidnull_destroy
Description: Destroy a null implementation of the spheroid component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_spheroidnull_dump
Description: Dump the contents of a null implementation of the spheroid component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_spheroidnull_dump_raw
Description: Dump the contents of a null implementation of the spheroid component in binary.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_spheroidnull_dump_xml
Description: Dump the contents of a null implementation of the spheroid component to XML.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_spheroidnull_initializor
Description: Initialize a null implementation of the spheroid component.
Code lines: 7
Contained by: module galacticus_nodes
19. Source Code Documentation

Modules used: memory_management

function: node_component_spheroidnull_is_active
Description: Return true if the null implementation of the spheroid component is the active choice.
Code lines: 6
contained by: module galacticus_nodes

subroutine: node_component_spheroidnull_name_from_index
Description: Return the name of the property of given index for a null implementation of the spheroid component.
Code lines: 9
Contains by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_spheroidnull_ode_step_rates_init
Description: Initialize rates in a null implementation of the spheroid component for an ODE solver step.
Code lines: 6
Contains by: module galacticus_nodes

subroutine: node_component_spheroidnull_odestepscalesinit
Description: Initialize scales in a null implementation of the spheroid component for an ODE solver step.
Code lines: 5
Contains by: module galacticus_nodes

subroutine: node_component_spheroidnull_output_count
Description: Increment output property count for a null implementation of the spheroid component.
Code lines: 9
Contains by: module galacticus_nodes

subroutine: node_component_spheroidnull_output_names
Description: Establish property names for a null implementation of the spheroid component.
Code lines: 11
Contains by: module galacticus_nodes

subroutine: node_component_spheroidnull_read_raw
Description: Read the contents of a null implementation of the spheroid component in binary.
Code lines: 8
Contains by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_spheroidnull_serialize_rates
Description: Serialize rates of a null implementation of the spheroid component.
Code lines: 7
Contains by: module galacticus_nodes

subroutine: node_component_spheroidnull_serialize_scales
Description: Serialize scales of a null implementation of the spheroid component.
Code lines: 7
Contains by: module galacticus_nodes
19.1. Program units

**subroutine**: node_component_spheroidnull_serialize_values
*Description*: Serialize values of a null implementation of the spheroid component.
*Code lines*: 7
*Contained by*: module galacticus_nodes

**function**: node_component_spheroidnull_type
*Description*: Returns the type for the null implementation of the spheroid component.
*Code lines*: 8
*Contained by*: module galacticus_nodes

**subroutine**: node_component_spheroidstandard_builder
*Description*: Build a standard implementation of the spheroid component.
*Code lines*: 79
*Contained by*: module galacticus_nodes
*Modules used*: fox_dom, galacticus_error, memory_management

**function**: node_component_spheroidstandard_count
*Description*: Return a count of the serialization of a standard implementation of the spheroid component.
*Code lines*: 13
*Contained by*: module galacticus_nodes

**subroutine**: node_component_spheroidstandard_deserialize_rates
*Description*: Serialize rates of a standard implementation of the spheroid component.
*Code lines*: 30
*Contained by*: module galacticus_nodes

**subroutine**: node_component_spheroidstandard_deserialize_scales
*Description*: Serialize scales of a standard implementation of the spheroid component.
*Code lines*: 30
*Contained by*: module galacticus_nodes

**subroutine**: node_component_spheroidstandard_deserialize_values
*Description*: Serialize values of a standard implementation of the spheroid component.
*Code lines*: 30
*Contained by*: module galacticus_nodes

**subroutine**: node_component_spheroidstandard_destroy
*Description*: Destroy a standard implementation of the spheroid component.
*Code lines*: 22
*Contained by*: module galacticus_nodes
*Modules used*: memory_management

**subroutine**: node_component_spheroidstandard_dump
*Description*: Dump the contents of a standard implementation of the spheroid component.
*Code lines*: 51
*Contained by*: module galacticus_nodes
19. Source Code Documentation

Modules used: galacticus_display
string_handling
iso_varying_string

subroutine: node_component_spheroidstandard_dump_raw
Description: Dump the contents of a standard implementation of the spheroid component in binary.
Code lines: 18
Contained by: module galacticus_nodes

subroutine: node_component_spheroidstandard_dump_xml
Description: Dump the contents of a standard implementation of the spheroid component to XML.
Code lines: 29
Contained by: module galacticus_nodes

subroutine: node_component_spheroidstandard_initializor
Description: Initialize a standard implementation of the spheroid component.
Code lines: 18
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_spheroidstandard_is_active
Description: Return true if the standard implementation of the spheroid component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_spheroidstandard_name_from_index
Description: Return the name of the property of given index for a standard implementation of the spheroid component.
Code lines: 49
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_spheroidstandard_ode_step_rates_init
Description: Initialize rates in a standard implementation of the spheroid component for an ODE solver step.
Code lines: 14
Contained by: module galacticus_nodes

subroutine: node_component_spheroidstandard_odestepsscalesinit
Description: Initialize scales in a standard implementation of the spheroid component for an ODE solver step.
Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_spheroidstandard_output_count
Description: Increment output property count for a standard implementation of the spheroid component.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: node_component_spheroidstandard_output_names
19.1. Program units

Description: Establish property names for a standard implementation of the spheroid component.
Code lines: 99
Contained by: module galacticus_nodes

subroutine: node_component_spheroidstandard_read_raw
Description: Read the contents of a standard implementation of the spheroid component in binary.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_spheroidstandard_serialize_rates
Description: Serialize rates of a standard implementation of the spheroid component.
Code lines: 30
Contained by: module galacticus_nodes

subroutine: node_component_spheroidstandard_serialize_scales
Description: Serialize scales of a standard implementation of the spheroid component.
Code lines: 30
Contained by: module galacticus_nodes

subroutine: node_component_spheroidstandard_serialize_values
Description: Serialize values of a standard implementation of the spheroid component.
Code lines: 30
Contained by: module galacticus_nodes

function: node_component_spheroidstandard_type
Description: Returns the type for the standard implementation of the spheroid component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_spin_builder
Description: Build a generic spin component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error

subroutine: node_component_spin_destroy
Description: Destroys a spin component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_spin_dump
Description: Dump the contents of a generic spin component.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string

subroutine: node_component_spin_initializor
Description: Initialize a generic spin component.
19. Source Code Documentation

Code lines: 8
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_spin_move
Description: Move instances of the spin component, from one node to another.

Code lines: 95
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_spin_output
Description: Output properties for a spin component.

Code lines: 23
Contained by: module galacticus_nodes

subroutine: node_component_spin_output_count
Description: Increment the count of properties to output for a generic spin component.

Code lines: 13
Contained by: module galacticus_nodes

subroutine: node_component_spin_output_names
Description: Establish property names for a generic spin component.

Code lines: 15
Contained by: module galacticus_nodes

subroutine: node_component_spin_remove
Description: Removes an instance of the spin component, shifting other instances to keep the array contiguous.

Code lines: 55
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: node_component_spin_type
Description: Returns the type for the spin component.

Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_spinnull_builder
Description: Build a null implementation of the spin component.

Code lines: 10
Contained by: module galacticus_nodes
Modules used: fox_dom, memory_management, galacticus_error

function: node_component_spinnull_count
Description: Return a count of the serialization of a null implementation of the spin component.

Code lines: 7
Contained by: module galacticus_nodes
subroutine: node_component_spinnull_deserialize_rates
Description: Serialize rates of a null implementation of the spin component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_spinnull_deserialize_scales
Description: Serialize scales of a null implementation of the spin component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_spinnull_deserialize_values
Description: Serialize values of a null implementation of the spin component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_spinnull_destroy
Description: Destroy a null implementation of the spin component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_spinnull_dump
Description: Dump the contents of a null implementation of the spin component.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_spinnull_dump_raw
Description: Dump the contents of a null implementation of the spin component in binary.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_spinnull_dump_xml
Description: Dump the contents of a null implementation of the spin component to XML.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_spinnull_initializor
Description: Initialize a null implementation of the spin component.
Code lines: 7
Contained by: module galacticus_nodes
Modules used: memory_management

function: node_component_spinnull_is_active
Description: Return true if the null implementation of the spin component is the active choice.
Code lines: 6
Contained by: module galacticus_nodes
**source code documentation**

**subroutine:** node_component_spinnull_name_from_index  
*Description:* Return the name of the property of given index for a null implementation of the spin component.  
*Code lines:* 9  
*Contained by:* module galacticus_nodes  
*Modules used:* iso_varying_string

**subroutine:** node_component_spinnull_ode_step_rates_init  
*Description:* Initialize rates in a null implementation of the spin component for an ODE solver step.  
*Code lines:* 6  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_spinnull_odestepscalesinit  
*Description:* Initialize scales in a null implementation of the spin component for an ODE solver step.  
*Code lines:* 5  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_spinnull_output_count  
*Description:* Increment output property count for a null implementation of the spin component.  
*Code lines:* 9  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_spinnull_output_names  
*Description:* Establish property names for a null implementation of the spin component.  
*Code lines:* 11  
*contained by:* module galacticus_nodes

**subroutine:** node_component_spinnull_read_raw  
*Description:* Read the contents of a null implementation of the spin component in binary.  
*Code lines:* 8  
*Contained by:* module galacticus_nodes  
*Modules used:* memory_management

**subroutine:** node_component_spinnull_serialize_rates  
*Description:* Serialize rates of a null implementation of the spin component.  
*Code lines:* 7  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_spinnull_serialize_scales  
*Description:* Serialize scales of a null implementation of the spin component.  
*Code lines:* 7  
*Contained by:* module galacticus_nodes

**subroutine:** node_component_spinnull_serialize_values  
*Description:* Serialize values of a null implementation of the spin component.  
*Code lines:* 7  
*Contained by:* module galacticus_nodes
19.1. Program units

function: node_component_spinnull_type
Description: Returns the type for the null implementation of the spin component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset3d_builder
Description: Build a preset3D implementation of the spin component.
Code lines: 33
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_spinpreset3d_count
Description: Return a count of the serialization of a preset3D implementation of the spin component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset3d_deserialize_rates
Description: Serialize rates of a preset3D implementation of the spin component.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset3d_deserialize_scales
Description: Serialize scales of a preset3D implementation of the spin component.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset3d_deserialize_values
Description: Serialize values of a preset3D implementation of the spin component.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset3d_destroy
Description: Destroy a preset3D implementation of the spin component.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_spinpreset3d_dump
Description: Dump the contents of a preset3D implementation of the spin component.
Code lines: 29
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_spinpreset3d_dump_raw
Description: Dump the contents of a preset3D implementation of the spin component in binary.
Code lines: 18
19. Source Code Documentation

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_spinpreset3d_dump_xml`

*Description:* Dump the contents of a preset3D implementation of the spin component to XML.

*Code lines:* 17

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_spinpreset3d_initializor`

*Description:* Initialize a preset3D implementation of the spin component.

*Code lines:* 13

*Contained by:* module `galacticus_nodes`

*Modules used:* `memory_management`

**function:** `node_component_spinpreset3d_is_active`

*Description:* Return true if the preset3D implementation of the spin component is the active choice.

*Code lines:* 6

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_spinpreset3d_name_from_index`

*Description:* Return the name of the property of given index for a preset3D implementation of the spin component.

*Code lines:* 16

*Contained by:* module `galacticus_nodes`

*Modules used:* `iso_varying_string`

**subroutine:** `node_component_spinpreset3d_ode_step_rates_init`

*Description:* Initialize rates in a preset3D implementation of the spin component for an ODE solver step.

*Code lines:* 8

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_spinpreset3d_odestepscalesinit`

*Description:* Initialize scales in a preset3D implementation of the spin component for an ODE solver step.

*Code lines:* 7

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_spinpreset3d_output_count`

*Description:* Increment output property count for a preset3D implementation of the spin component.

*Code lines:* 11

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_spinpreset3d_output_names`

*Description:* Establish property names for a preset3D implementation of the spin component.

*Code lines:* 33

*Contained by:* module `galacticus_nodes`

**subroutine:** `node_component_spinpreset3d_read_raw`

*Description:* Read the contents of a preset3D implementation of the spin component in binary.

*Code lines:* 25

*Contained by:* module `galacticus_nodes`
19.1. Program units

Modules used: memory_management

subroutine: node_component_spinpreset3d_serialize_rates
Description: Serialize rates of a preset3D implementation of the spin component.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset3d_serialize_scales
Description: Serialize scales of a preset3D implementation of the spin component.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset3d_serialize_values
Description: Serialize values of a preset3D implementation of the spin component.
Code lines: 19
Contained by: module galacticus_nodes

function: node_component_spinpreset3d_type
Description: Returns the type for the preset3D implementation of the spin component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset_builder
Description: Build a preset implementation of the spin component.
Code lines: 25
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_spinpreset_count
Description: Return a count of the serialization of a preset implementation of the spin component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset_deserialize_rates
Description: Serialize rates of a preset implementation of the spin component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset_deserialize_scales
Description: Serialize scales of a preset implementation of the spin component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset_deserialize_values
Description: Serialize values of a preset implementation of the spin component.
Code lines: 11
Contained by: module galacticus_nodes
subroutine: node_component_spinpreset_destroy
  Description: Destroy a preset implementation of the spin component.
  Code lines: 7
  Contained by: module galacticus_nodes
  Modules used: memory_management

subroutine: node_component_spinpreset_dump
  Description: Dump the contents of a preset implementation of the spin component.
  Code lines: 19
  Contained by: module galacticus_nodes
  Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_spinpreset_dump_raw
  Description: Dump the contents of a preset implementation of the spin component in binary.
  Code lines: 9
  Contained by: module galacticus_nodes

subroutine: node_component_spinpreset_dump_xml
  Description: Dump the contents of a preset implementation of the spin component to XML.
  Code lines: 11
  Contained by: module galacticus_nodes

subroutine: node_component_spinpreset_initializor
  Description: Initialize a preset implementation of the spin component.
  Code lines: 9
  Contained by: module galacticus_nodes
  Modules used: memory_management

function: node_component_spinpreset_is_active
  Description: Return true if the preset implementation of the spin component is the active choice.
  Code lines: 6
  Contained by: module galacticus_nodes

subroutine: node_component_spinpreset_name_from_index
  Description: Return the name of the property of given index for a preset implementation of the spin component.
  Code lines: 14
  Contained by: module galacticus_nodes
  Modules used: iso_varying_string

subroutine: node_component_spinpreset_ode_step_rates_init
  Description: Initialize rates in a preset implementation of the spin component for an ODE solver step.
  Code lines: 7
  Contained by: module galacticus_nodes

subroutine: node_component_spinpreset_odestepscalesinit
  Description: Initialize scales in a preset implementation of the spin component for an ODE solver step.
19.1. Program units

Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset_output_count
Description: Increment output property count for a preset implementation of the spin component.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset_output_names
Description: Establish property names for a preset implementation of the spin component.
Code lines: 24
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset_read_raw
Description: Read the contents of a preset implementation of the spin component in binary.
Code lines: 10
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_spinpreset_serialize_rates
Description: Serialize rates of a preset implementation of the spin component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset_serialize_scales
Description: Serialize scales of a preset implementation of the spin component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_spinpreset_serialize_values
Description: Serialize values of a preset implementation of the spin component.
Code lines: 11
Contained by: module galacticus_nodes

function: node_component_spinpreset_type
Description: Returns the type for the preset implementation of the spin component.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_spinrandom_builder
Description: Build a random implementation of the spin component.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: fox_dom galacticus_error memory_management

function: node_component_spinrandom_count
Description: Return a count of the serialization of a random implementation of the spin component.
Code lines: 8
19. Source Code Documentation

Contained by: module galacticus_nodes

**subroutine:** node_component_spinrandom_deserialize_rates
*Description:* Serialize rates of a random implementation of the spin component.
*Code lines:* 11
*Contained by:* module galacticus_nodes

**subroutine:** node_component_spinrandom_deserialize_scales
*Description:* Serialize scales of a random implementation of the spin component.
*Code lines:* 11
*Contained by:* module galacticus_nodes

**subroutine:** node_component_spinrandom_deserialize_values
*Description:* Serialize values of a random implementation of the spin component.
*Code lines:* 11
*Contained by:* module galacticus_nodes

**subroutine:** node_component_spinrandom_destroy
*Description:* Destroy a random implementation of the spin component.
*Code lines:* 7
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**subroutine:** node_component_spinrandom_dump
*Description:* Dump the contents of a random implementation of the spin component.
*Code lines:* 16
*Contained by:* module galacticus_nodes
*Modules used:* galacticus_display, iso_varying_string, string_handling

**subroutine:** node_component_spinrandom_dump_raw
*Description:* Dump the contents of a random implementation of the spin component in binary.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine:** node_component_spinrandom_dump_xml
*Description:* Dump the contents of a random implementation of the spin component to XML.
*Code lines:* 11
*Contained by:* module galacticus_nodes

**subroutine:** node_component_spinrandom_initializor
*Description:* Initialize a random implementation of the spin component.
*Code lines:* 8
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**function:** node_component_spinrandom_is_active
*Description:* Return true if the random implementation of the spin component is the active choice.
*Code lines:* 6
19.1. Program units

Contained by: module galacticus_nodes

subroutine: node_component_spinrandom_name_from_index
Description: Return the name of the property of given index for a random implementation of the spin component.
Code lines: 14
Contained by: module galacticus_nodes
Modules used: iso_varying_string

subroutine: node_component_spinrandom_ode_step_rates_init
Description: Initialize rates in a random implementation of the spin component for an ODE solver step.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_spinrandom_odestepscalesinit
Description: Initialize scales in a random implementation of the spin component for an ODE solver step.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: node_component_spinrandom_output_count
Description: Increment output property count for a random implementation of the spin component.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_component_spinrandom_output_names
Description: Establish property names for a random implementation of the spin component.
Code lines: 24
Contained by: module galacticus_nodes

subroutine: node_component_spinrandom_read_raw
Description: Read the contents of a random implementation of the spin component in binary.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: node_component_spinrandom_serialize_rates
Description: Serialize rates of a random implementation of the spin component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_spinrandom_serialize_scales
Description: Serialize scales of a random implementation of the spin component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_spinrandom_serialize_values
Description: Serialize values of a random implementation of the spin component.
Code lines: 11
Contained by: module galacticus_nodes
function: node_component_spinrandom_type
Description: Returns the type for the random implementation of the spin component.
Code lines: 8
Contained by: module galacticus_nodes

function: node_component_surface_density_null
Description: A null implementation of the surface density in a component. Always returns zero.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: node_dump
Description: Dump node content.
Code lines: 131
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_dump_raw
Description: Dump node content in binary.
Code lines: 363
Contained by: module galacticus_nodes

subroutine: node_dump_xml
Description: Dump node content.
Code lines: 126
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_output
Code lines: 101
Contained by: module galacticus_nodes

subroutine: node_output_count
Description: Increment the count of properties to output for this node.
Code lines: 99
Contained by: module galacticus_nodes

subroutine: node_output_names
Description: Establish the names of properties to output for this node.
Code lines: 101
Contained by: module galacticus_nodes

function: node_property_name_from_index
Description: Return the name of a property given its index.
Code lines: 241
Contained by: module galacticus_nodes
19.1. Program units

Modules used: iso_varying_string

subroutine: node_read_raw
Description: Dump node content in binary.
Code lines: 544
Contained by: module galacticus_nodes

type: nodecomponent
Description: A class for components in nodes.
Code lines: 204
Contained by: module galacticus_nodes

type: nodecomponentagestatistics
Description: Type for the ageStatistics component class.
Code lines: 265
Contained by: module galacticus_nodes

type: nodecomponentagestatisticsnull
Description: Class for the null implementation of the ageStatistics component.
Code lines: 31
Contained by: module galacticus_nodes

type: nodecomponentagestatisticsstandard
Description: Class for the standard implementation of the ageStatistics component.
Code lines: 113
Contained by: module galacticus_nodes

type: nodecomponentbasic
Description: Type for the basic component class.
Code lines: 272
Contained by: module galacticus_nodes

type: nodecomponentbasicnonevolving
Description: Class for the nonEvolving implementation of the basic component.
Code lines: 81
Contained by: module galacticus_nodes

type: nodecomponentbasicnull
Description: Class for the null implementation of the basic component.
Code lines: 31
Contained by: module galacticus_nodes

type: nodecomponentbasicstandard
Description: Class for the standard implementation of the basic component.
Code lines: 101
Contained by: module galacticus_nodes

type: nodecomponentbasicstandardtracking
Description: Class for the standardTracking implementation of the basic component.
19. Source Code Documentation

Code lines: 50
Contained by: module galacticus_nodes

type: nodecomponentblackhole
Description: Type for the blackHole component class.
Code lines: 363
Contained by: module galacticus_nodes

type: nodecomponentblackholenull
Description: Class for the null implementation of the blackHole component.
Code lines: 31
Contained by: module galacticus_nodes

type: nodecomponentblackholesimple
Description: Class for the simple implementation of the blackHole component.
Code lines: 57
Contained by: module galacticus_nodes

type: nodecomponentblackholestandard
Description: Class for the standard implementation of the blackHole component.
Code lines: 145
Contained by: module galacticus_nodes

type: nodecomponentdarkmatterprofile
Description: Type for the darkMatterProfile component class.
Code lines: 237
Contained by: module galacticus_nodes

type: nodecomponentdarkmatterprofilenull
Description: Class for the null implementation of the darkMatterProfile component.
Code lines: 31
Contained by: module galacticus_nodes

type: nodecomponentdarkmatterprofilescalescale
Description: Class for the scale implementation of the darkMatterProfile component.
Code lines: 85
Contained by: module galacticus_nodes

type: nodecomponentdarkmatterprofilescalescaleshape
Description: Class for the scaleShape implementation of the darkMatterProfile component.
Code lines: 70
Contained by: module galacticus_nodes

type: nodecomponentdisk
19.1. Program units

**Description:** Type for the disk component class.
**Code lines:** 657
**Contained by:** module `galacticus_nodes`  

**type: nodecomponentdiskexponential**
**Description:** Class for the exponential implementation of the disk component.
**Code lines:** 277
**Contained by:** module `galacticus_nodes`  

**type: nodecomponentdisknull**
**Description:** Class for the null implementation of the disk component.
**Code lines:** 31
**Contained by:** module `galacticus_nodes`  

**type: nodecomponentdiskverysimple**
**Description:** Class for the verySimple implementation of the disk component.
**Code lines:** 81
**Contained by:** module `galacticus_nodes`  

**type: nodecomponentdynamicsstatistics**
**Description:** Type for the `dynamicsStatistics` component class.
**Code lines:** 146
**Contained by:** module `galacticus_nodes`  

**type: nodecomponentdynamicsstatisticsbars**
**Description:** Class for the bars implementation of the `dynamicsStatistics` component.
**Code lines:** 91
**Contained by:** module `galacticus_nodes`  

**type: nodecomponentdynamicsstatisticsnull**
**Description:** Class for the null implementation of the `dynamicsStatistics` component.
**Code lines:** 31
**Contained by:** module `galacticus_nodes`  

**type: nodecomponentformationtime**
**Description:** Type for the `formationTime` component class.
**Code lines:** 69
**Contained by:** module `galacticus_nodes`  

**type: nodecomponentformationtimecole2000**
**Description:** Class for the Cole2000 implementation of the `formationTime` component.
**Code lines:** 35
**Contained by:** module `galacticus_nodes`  

**type: nodecomponentformationtimenull**
**Description:** Class for the null implementation of the `formationTime` component.
**Code lines:** 31
**Contained by:** module `galacticus_nodes`
type: nodecomponenthosthistory
Description: Type for the hostHistory component class.
Code lines: 76
Contained by: module galacticus_nodes

type: nodecomponenthosthistorynull
Description: Class for the null implementation of the hostHistory component.
Code lines: 31
Contained by: module galacticus_nodes

type: nodecomponenthosthistorystandard
Description: Class for the standard implementation of the hostHistory component.
Code lines: 50
Contained by: module galacticus_nodes

type: nodecomponenthothalo
Description: Type for the hotHalo component class.
Code lines: 1350
Contained by: module galacticus_nodes

type: nodecomponenthothalocoldmode
Description: Class for the coldMode implementation of the hotHalo component.
Code lines: 135
Contained by: module galacticus_nodes

type: nodecomponenthothalonull
Description: Class for the null implementation of the hotHalo component.
Code lines: 31
Contained by: module galacticus_nodes

type: nodecomponenthothalooutflowtracking
Description: Class for the outflowTracking implementation of the hotHalo component.
Code lines: 116
Contained by: module galacticus_nodes

type: nodecomponenthothalostandard
Description: Class for the standard implementation of the hotHalo component.
Code lines: 462
Contained by: module galacticus_nodes

type: nodecomponenthothaloverysimple
Description: Class for the verySimple implementation of the hotHalo component.
Code lines: 95
Contained by: module galacticus_nodes

type: nodecomponentindices
Description: Type for the indices component class.
Code lines: 76
19.1. Program units

```
contained by: module galacticus_nodes

type: nodecomponentindicesnull
  description: Class for the null implementation of the indices component.
  code lines: 31
  contained by: module galacticus_nodes

type: nodecomponentindicesstandard
  description: Class for the standard implementation of the indices component.
  code lines: 50
  contained by: module galacticus_nodes

type: nodecomponentinteroutput
  description: Type for the interOutput component class.
  code lines: 153
  contained by: module galacticus_nodes

type: nodecomponentinteroutputnull
  description: Class for the null implementation of the interOutput component.
  code lines: 31
  contained by: module galacticus_nodes

type: nodecomponentinteroutputstandard
  description: Class for the standard implementation of the interOutput component.
  code lines: 73
  contained by: module galacticus_nodes

type: nodecomponentmassflowstatistics
  description: Type for the massFlowStatistics component class.
  code lines: 97
  contained by: module galacticus_nodes

type: nodecomponentmassflowstatisticsnull
  description: Class for the null implementation of the massFlowStatistics component.
  code lines: 31
  contained by: module galacticus_nodes

type: nodecomponentmassflowstatisticsstandard
  description: Class for the standard implementation of the massFlowStatistics component.
  code lines: 53
  contained by: module galacticus_nodes

type: nodecomponentmergingstatistics
  description: Type for the mergingStatistics component class.
  code lines: 223
  contained by: module galacticus_nodes

type: nodecomponentmergingstatisticsnull
  description: Class for the null implementation of the mergingStatistics component.
```
19. Source Code Documentation

Code lines: 31  Contained by: module galacticus_nodes

type: nodecomponentmergingstatisticsrecent
Description: Class for the recent implementation of the mergingStatistics component.
Code lines: 50  Contained by: module galacticus_nodes

Code lines: 101  Contained by: module galacticus_nodes

type: nodecomponentmergingstatisticsstandard
Description: Class for the standard implementation of the mergingStatistics component.
Code lines: 101  Contained by: module galacticus_nodes

Code lines: 146  Contained by: module galacticus_nodes

type: nodecomponentnbody
Description: Type for the nBody component class.
Code lines: 146  Contained by: module galacticus_nodes

Code lines: 31  Contained by: module galacticus_nodes

type: nodecomponentnbodynull
Description: Class for the null implementation of the nBody component.
Code lines: 31  Contained by: module galacticus_nodes

Code lines: 84  Contained by: module galacticus_nodes

type: nodecomponentnbodystandard
Description: Class for the standard implementation of the nBody component.
Code lines: 84  Contained by: module galacticus_nodes

Code lines: 146  Contained by: module galacticus_nodes

Code lines: 31  Contained by: module galacticus_nodes

type: nodecomponentposition
Description: Type for the position component class.
Code lines: 31  Contained by: module galacticus_nodes

Code lines: 84  Contained by: module galacticus_nodes

Code lines: 566  Contained by: module galacticus_nodes

type: nodecomponentpositionnull
Description: Class for the null implementation of the position component.
Code lines: 72  Contained by: module galacticus_nodes

Code lines: 31  Contained by: module galacticus_nodes

Code lines: 72  Contained by: module galacticus_nodes

type: nodecomponentsatellite
Description: Type for the satellite component class.
Code lines: 566  Contained by: module galacticus_nodes

Code lines: 1622
19.1. Program units

**Description:** Class for the null implementation of the satellite component.
**Code lines:** 31
**Contained by:** module `galacticus_nodes`

**type: nodecomponentsatelliteorbiting**

**Description:** Class for the orbiting implementation of the satellite component.
**Code lines:** 170
**Contained by:** module `galacticus_nodes`

**type: nodecomponentsatellitepreset**

**Description:** Class for the preset implementation of the satellite component.
**Code lines:** 112
**Contained by:** module `galacticus_nodes`

**type: nodecomponentsatellitestandard**

**Description:** Class for the standard implementation of the satellite component.
**Code lines:** 117
**Contained by:** module `galacticus_nodes`

**type: nodecomponentsatelliteverysimple**

**Description:** Class for the verySimple implementation of the satellite component.
**Code lines:** 50
**Contained by:** module `galacticus_nodes`

**type: nodecomponentspheroid**

**Description:** Type for the `spheroid` component class.
**Code lines:** 734
**Contained by:** module `galacticus_nodes`

**type: nodecomponentspheroidnull**

**Description:** Class for the null implementation of the spheroid component.
**Code lines:** 31
**Contained by:** module `galacticus_nodes`

**type: nodecomponentspheroidstandard**

**Description:** Class for the standard implementation of the spheroid component.
**Code lines:** 322
**Contained by:** module `galacticus_nodes`

**type: nodecomponentspin**

**Description:** Type for the `spin` component class.
**Code lines:** 237
**Contained by:** module `galacticus_nodes`

**type: nodecomponentspinnull**

**Description:** Class for the null implementation of the spin component.
**Code lines:** 31
**Contained by:** module `galacticus_nodes`
type: nodecomponentspinpreset
  Description: Class for the preset implementation of the spin component.
  Code lines: 70
  Contained by: module galacticus_nodes

type: nodecomponentspinpreset3d
  Description: Class for the preset3D implementation of the spin component.
  Code lines: 70
  Contained by: module galacticus_nodes

type: nodecomponentspinrandom
  Description: Class for the random implementation of the spin component.
  Code lines: 56
  Contained by: module galacticus_nodes

type: nodedataabundancesscalarevolvable
  Description: Type describing an evolvable scalar abundance property of a node component.
  Code lines: 3
  Contained by: module galacticus_nodes

type: nodedatachemicalabundancesscalarevolvable
  Description: Type describing an evolvable scalar chemicalAbundances property of a node component.
  Code lines: 3
  Contained by: module galacticus_nodes

type: nodedatadouble1d
  Description: Type describing a non-evolvable 1-D double property of a node component.
  Code lines: 3
  Contained by: module galacticus_nodes

type: nodedatadouble1devolvable
  Description: Type describing an evolvable 1-D double property of a node component.
  Code lines: 3
  Contained by: module galacticus_nodes

type: nodedatadoublescalar
  Description: Type describing a non-evolvable scalar double property of a node component.
  Code lines: 3
  Contained by: module galacticus_nodes

type: nodedatadoublescalarevolvable
  Description: Type describing an evolvable scalar double property of a node component.
  Code lines: 3
  Contained by: module galacticus_nodes

type: nodedatahistoryscalar


19.1. Program units

Description: Type describing an non-evolvable scalar normalfont history property of a node component.
Code lines: 3
Contained by: module galacticus_nodes

type: nodedatahistoryscalarevolvable
Description: Type describing an evolvable scalar normalfont history property of a node component.
Code lines: 3
Contained by: module galacticus_nodes

type: nodedatainteger1d
Description: Type describing a non-evolvable 1-D integer property of a node component.
Code lines: 3
Contained by: module galacticus_nodes

type: nodedataintegerscalar
Description: Type describing a non-evolvable scalar integer property of a node component.
Code lines: 3
Contained by: module galacticus_nodes

type: nodedatakeplerorbitscalar
Description: Type describing a non-evolvable scalar keplerOrbit property of a node component.
Code lines: 3
Contained by: module galacticus_nodes

type: nodedatalogicalscalar
Description: Type describing a non-evolvable scalar logical property of a node component.
Code lines: 3
Contained by: module galacticus_nodes

type: nodedatalongintegerhistoryscalarevolvable
Description: Type describing an non-evolvable scalar longIntegerHistory property of a node component.
Code lines: 3
Contained by: module galacticus_nodes

type: nodedatalongintegerscalar
Description: Type describing a non-evolvable scalar long integer property of a node component.
Code lines: 3
Contained by: module galacticus_nodes

type: nodedatastellarluminositiesscalarevolvable
Description: Type describing an evolvable scalar stellarLuminosities property of a node component.
Code lines: 3
Contained by: module galacticus_nodes

type: nodedatatensorrank2dimension3symmetricscalarevolvable
19. Source Code Documentation

**Description:** Type describing an evolvable scalar normalfont tensorRank2Dimension3Symmetric property of a node component.

**Contained by:** module galacticus_nodes

**type:** nodeevent

**Description:** Type for events attached to nodes.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**function:** positioncountlinked

**Description:** Returns the number of position components in self.

**Code lines:** 19

**Contained by:** module galacticus_nodes

**subroutine:** positioncreatebyinterrupt

**Description:** Create the position component of self via an interrupt.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**subroutine:** positioncreatelinked

**Description:** Create the position component of self.

**Code lines:** 29

**Modules used:**
galacticus_display
iso_varying_string
string_handling

**subroutine:** positiondestroylinked

**Description:** Destroy the position component of self.

**Code lines:** 13

**Contained by:** module galacticus_nodes

**function:** positionget

**Description:** Returns the position component of self.

**Code lines:** 27

**Contained by:** module galacticus_nodes

**Modules used:**
galacticus_error

**function:** positionnullbindingdouble1inout

**Description:** A null get function for rank 1 double precisions.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**function:** positionnullbindinghistory0inout

**Description:** A null get function for rank 0 typehistorys.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**function:** positionnullbindinginteger0in
19.1. Program units

Description: A null get function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: positionnullbindingratedouble1inout
Description: A null rate function for rank 1 double precisions.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: positionnullbindingratehistory0inout
Description: A null rate function for rank 0 typehistorys.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: positionnullbindingrateinteger0in
Description: A null rate function for rank 0 integers.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: positionnullbindingsetdouble1inout
Description: A null set function for rank 1 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: positionnullbindingsethistory0inout
Description: A null set function for rank 0 typehistorys.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: positionnullbindingsetinteger0in
Description: A null set function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

function: positionposition
Description: Returns the default value for the position property for the position component class.
Code lines: 13
Contained by: module galacticus_nodes

function: positionpositionattributematch
Description: Return a text list of component implementations in the position class that have the desired attributes for the position property.
Code lines: 27
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: positionpositionhistory
Description: Returns the default value for the positionHistory property for the position component class.
Code lines: 8
**function: positionpositionhistoryattributematch**

*Description:* Return a text list of component implementations in the `position` class that have the desired attributes for the `positionHistory` property.

*Code lines:* 27

*Contained by:* module `galacticus_nodes`

*Modules used:* `iso_varying_string`

**function: positionpositionhistoryisgettable**

*Description:* Returns true if the `positionHistory` property is gettable for the `position` component class.

*Code lines:* 6

*Contained by:* module `galacticus_nodes`

**function: positionpositionisgettable**

*Description:* Returns true if the `position` property is gettable for the `position` component class.

*Code lines:* 7

*Contained by:* module `galacticus_nodes`

**function: positionpresetposition**

*Description:* Return the position of the node.

*Code lines:* 32

*Contained by:* module `galacticus_nodes`

*Modules used:* `fgsl`, `iso_c_binding`, `numerical_interpolation`

**function: positionpresetpositionhistoryget**

*Description:* Return the `positionHistory` property of the `PositionPreset` component implementation.

*Code lines:* 8

*Contained by:* module `galacticus_nodes`

**subroutine: positionpresetpositionhistoryset**

*Description:* Set the `positionHistory` property of the `PositionPreset` component implementation.

*Code lines:* 9

*Contained by:* module `galacticus_nodes`

*Modules used:* `memory_management`

**subroutine: positionpresetpositionset**

*Description:* Set the `position` property of the `PositionPreset` component implementation.

*Code lines:* 17

*Contained by:* module `galacticus_nodes`

*Modules used:* `memory_management`

**function: positionpresetvelocity**

*Description:* Return the velocity of the node.

*Code lines:* 32

*Contained by:* module `galacticus_nodes`

*Modules used:* `fgsl`, `iso_c_binding`, `numerical_interpolation`
**subroutine: positionpresetvelocityset**

*Description:* Set the *velocity* property of the *PositionPreset* component implementation.

*Code lines:* 17

*Contained by:* module `galacticus_nodes`

*Modules used:* `memory_management`

**function: positionvelocity**

*Description:* Returns the default value for the *velocity* property for the *position* component class.

*Code lines:* 13

*Contained by:* module `galacticus_nodes`

**function: positionvelocityattributematch**

*Description:* Return a text list of component implementations in the *position* class that have the desired attributes for the *velocity* property.

*Code lines:* 27

*Contained by:* module `galacticus_nodes`

*Modules used:* `iso_varying_string`

**function: positionvelocityisgettable**

*Description:* Returns true if the *velocity* property is gettable for the *position* component class.

*Code lines:* 7

*Contained by:* module `galacticus_nodes`

**function: satelliteboundmass**

*Description:* Returns the default value for the *boundMass* property for the *satellite* component class.

*Code lines:* 22

*Contained by:* module `galacticus_nodes`

**function: satelliteboundmassattributematch**

*Description:* Return a text list of component implementations in the *satellite* class that have the desired attributes for the *boundMass* property.

*Code lines:* 45

*Contained by:* module `galacticus_nodes`

*Modules used:* `iso_varying_string`

**function: satelliteboundmasshistory**

*Description:* Returns the default value for the *boundMassHistory* property for the *satellite* component class.

*Code lines:* 8

*Contained by:* module `galacticus_nodes`

**function: satelliteboundmasshistoryattributematch**

*Description:* Return a text list of component implementations in the *satellite* class that have the desired attributes for the *boundMassHistory* property.

*Code lines:* 27

*Contained by:* module `galacticus_nodes`

*Modules used:* `iso_varying_string`

**function: satelliteboundmasshistoryisgettable**
Description: Returns true if the boundMassHistory property is gettable for the satellite component class.
Code lines: 6
Contained by: module galacticus_nodes

function: satelliteboundmassisgettable
Description: Returns true if the boundMass property is gettable for the satellite component class.
Code lines: 8
Contained by: module galacticus_nodes

function: satellitecountlinked
Description: Returns the number of satellite components in self.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: satellitecreatebyinterrupt
Description: Create the satellite component of self via an interrupt.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satellitecreatelinked
Description: Create the satellite component of self.
Code lines: 29
Contained by: module galacticus_nodes

subroutine: satellitedestroylinked
Description: Destroy the satellite component of self.
Code lines: 13
Contained by: module galacticus_nodes

function: satelliteget
Description: Returns the satellite component of self.
Code lines: 27
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

function: satellitemergetime
Description: Returns the default value for the mergeTime property for the satellite component class.
Code lines: 24
Contained by: module galacticus_nodes

function: satellitemergetimeattributematch
Description: Return a text list of component implementations in the satellite class that have the desired attributes for the mergeTime property
Code lines: 56
Contained by: module galacticus_nodes
Modules used: iso_varying_string
19.1. Program units

function: satellitemergetimeisgettable
Description: Returns true if the `mergeTime` property is gettable for the `satellite` component class.
Code lines: 10
Contained by: module `galacticus_nodes`

function: satellitenodeindex
Description: Returns the default value for the `nodeIndex` property for the `satellite` component class.
Code lines: 8
Contained by: module `galacticus_nodes`

function: satellitenodeindexattributematch
Description: Return a text list of component implementations in the `satellite` class that have the desired attributes for the `nodeIndex` property.
Code lines: 27
Contained by: module `galacticus_nodes`
Modules used: `iso_varying_string`

function: satellitenodeindexhistory
Description: Returns the default value for the `nodeIndexHistory` property for the `satellite` component class.
Code lines: 8
Contained by: module `galacticus_nodes`

function: satellitenodeindexhistoryattributematch
Description: Return a text list of component implementations in the `satellite` class that have the desired attributes for the `nodeIndexHistory` property.
Code lines: 27
Contained by: module `galacticus_nodes`
Modules used: `iso_varying_string`

function: satellitenodeindexhistoryisgettable
Description: Returns true if the `nodeIndexHistory` property is gettable for the `satellite` component class.
Code lines: 6
Contained by: module `galacticus_nodes`

function: satellitenodeindexisgettable
Description: Returns true if the `nodeIndex` property is gettable for the `satellite` component class.
Code lines: 6
Contained by: module `galacticus_nodes`

function: satellitenullbindingdouble0inout
Description: A null get function for rank 0 double precisions.
Code lines: 7
Contained by: module `galacticus_nodes`

function: satellitenullbindingdouble1inout
Description: A null get function for rank 1 double precisions.
Code lines: 7
19. Source Code Documentation

*Contained by:* module *galacticus_nodes*

**function:** satellitenullbindinghistory0inout

*Description:* A null get function for rank 0 typehistorys.

*Code lines:* 7

*Contained by:* module *galacticus_nodes*

**function:** satellitenullbindinginteger0in

*Description:* A null get function for rank 0 integers.

*Code lines:* 7

*Contained by:* module *galacticus_nodes*

**function:** satellitenullbindingkeplerorbit0inout

*Description:* A null get function for rank 0 typekeplerorbits.

*Code lines:* 7

*Contained by:* module *galacticus_nodes*

**function:** satellitenullbindinglonginteger0inout

*Description:* A null get function for rank 0 integer(kind=kind_int8)s.

*Code lines:* 7

*Contained by:* module *galacticus_nodes*

**function:** satellitenullbindinglongintegerhistory0inout

*Description:* A null get function for rank 0 typelongintegerhistorys.

*Code lines:* 7

*Contained by:* module *galacticus_nodes*

**subroutine:** satellitenullbindingratedouble0inout

*Description:* A null rate function for rank 0 double precisions.

*Code lines:* 9

*Contained by:* module *galacticus_nodes*

**subroutine:** satellitenullbindingratedouble1inout

*Description:* A null rate function for rank 1 double precisions.

*Code lines:* 9

*Contained by:* module *galacticus_nodes*

**subroutine:** satellitenullbindingratehistory0inout

*Description:* A null rate function for rank 0 typehistorys.

*Code lines:* 9

*Contained by:* module *galacticus_nodes*

**subroutine:** satellitenullbindingrateinteger0in

*Description:* A null rate function for rank 0 integers.

*Code lines:* 9

*Contained by:* module *galacticus_nodes*

**subroutine:** satellitenullbindingratekeplerorbit0inout

*Description:* A null rate function for rank 0 typekeplerorbits.
19.1. Program units

**subroutine:** satellitenullbindingratelonginteger0inout

*Description:* A null rate function for rank 0 integer(kind=kind_int8)s.

*Code lines:* 9
*Contained by:* module galacticus_nodes

**subroutine:** satellitenullbindingratelongintegerhistory0inout

*Description:* A null rate function for rank 0 typelongintegerhistories.

*Code lines:* 9
*Contained by:* module galacticus_nodes

**subroutine:** satellitenullbindingratetensordimension3symmetric0inout

*Description:* A null rate function for rank 0 typetensordimension3symmetrics.

*Code lines:* 9
*Contained by:* module galacticus_nodes

**subroutine:** satellitenullbindingsetdouble0inout

*Description:* A null set function for rank 0 double precisions.

*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** satellitenullbindingsetdouble1inout

*Description:* A null set function for rank 1 double precisions.

*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** satellitenullbindingsethistory0inout

*Description:* A null set function for rank 0 typehistories.

*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** satellitenullbindingsetinteger0in

*Description:* A null set function for rank 0 integers.

*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** satellitenullbindingsetkeplerorbit0inout

*Description:* A null set function for rank 0 typekeplerorbits.

*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** satellitenullbindingsetlonginteger0inout

*Description:* A null set function for rank 0 integer(kind=kind_int8)s.

*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine:** satellitenullbindingsetlongintegerhistory0inout
19. Source Code Documentation

Description: A null set function for rank 0 typelongintegerhistorys.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: satellitenullbindingsettensorrank2dimension3symmetric0inout
Description: A null set function for rank 0 typetensorrank2dimension3symmetrics.
Code lines: 7
Contained by: module galacticus_nodes

function: satellitenullbindingtensorrank2dimension3symmetric0inout
Description: A null get function for rank 0 typetensorrank2dimension3symmetrics.
Code lines: 7
Contained by: module galacticus_nodes

function: satelliteorbitingboundmasscount
Description: Return a count of the number of scalar properties in the boundMass property of the satelliteOrbiting component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: satelliteorbitingboundmassget
Description: Return the boundMass property of the SatelliteOrbiting component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satelliteorbitingboundmassrate
Description: Accumulate to the boundMass property rate of change of the SatelliteOrbiting component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: satelliteorbitingboundmassscale
Description: Set the boundMass property scale of the SatelliteOrbiting component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satelliteorbitingboundmassset
Description: Set the boundMass property of the SatelliteOrbiting component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: satelliteorbitingmergetimeget
Description: Return the mergeTime property of the SatelliteOrbiting component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satelliteorbitingmergetimeset
Description: Set the mergeTime property of the SatelliteOrbiting component implementation.
19.1. Program units

function: satelliteorbitingpositioncount
Description: Return a count of the number of scalar properties in the position property of the satelliteOrbiting component implementation.
Code lines: 9
Contained by: module galacticus_nodes

function: satelliteorbitingpositionget
Description: Return the position property of the SatelliteOrbiting component implementation.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: satelliteorbitingpositionrate
Description: Accumulate to the position property rate of change of the SatelliteOrbiting component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satelliteorbitingpositionscale
Description: Set the position property scale of the SatelliteOrbiting component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: satelliteorbitingpositionset
Description: Set the position property of the SatelliteOrbiting component implementation.
Code lines: 35
Contained by: module galacticus_nodes

function: satelliteorbitingtidalheatingnormalizedcount
Description: Return a count of the number of scalar properties in the tidalHeatingNormalized property of the satelliteOrbiting component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: satelliteorbitingtidalheatingnormalizedget
Description: Return the tidalHeatingNormalized property of the SatelliteOrbiting component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satelliteorbitingtidalheatingnormalizedrate
Description: Accumulate to the tidalHeatingNormalized property rate of change of the SatelliteOrbiting component implementation.
Code lines: 10
Contained by: module galacticus_nodes
subroutine: satelliteorbitingtidalheatingnormalizedscale
Description: Set the tidalHeatingNormalized property scale of the SatelliteOrbiting component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satelliteorbitingtidalheatingnormalizedset
Description: Set the tidalHeatingNormalized property of the SatelliteOrbiting component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: satelliteorbitingtidaltensorpathintegratedcount
Description: Return a count of the number of scalar properties in the tidalTensorPathIntegrated property of the SatelliteOrbiting component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: satelliteorbitingtidaltensorpathintegratedget
Description: Return the tidalTensorPathIntegrated property of the SatelliteOrbiting component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satelliteorbitingtidaltensorpathintegratedrate
Description: Accumulate to the tidalTensorPathIntegrated property rate of change of the SatelliteOrbiting component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: satelliteorbitingtidaltensorpathintegratedscale
Description: Set the tidalTensorPathIntegrated property scale of the SatelliteOrbiting component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satelliteorbitingtidaltensorpathintegratedset
Description: Set the tidalTensorPathIntegrated property of the SatelliteOrbiting component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: satelliteorbitingvelocitycount
Description: Return a count of the number of scalar properties in the velocity property of the SatelliteOrbiting component implementation.
Code lines: 11
Contained by: module galacticus_nodes
function: satelliteorbitingvelocityget
Description: Return the velocity property of the SatelliteOrbiting component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satelliteorbitingvelocityrate
Description: Accumulate to the velocity property rate of change of the SatelliteOrbiting component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: satelliteorbitingvelocityscale
Description: Set the velocity property scale of the SatelliteOrbiting component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satelliteorbitingvelocityset
Description: Set the velocity property of the SatelliteOrbiting component implementation.
Code lines: 35
Contained by: module galacticus_nodes
Modules used: memory_management

function: satelliteorbitingvirialorbitget
Description: Return the virialOrbit property of the SatelliteOrbiting component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satelliteorbitingvirialorbitset
Description: Set the virialOrbit property of the SatelliteOrbiting component implementation.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

function: satelliteposition
Description: Returns the default value for the position property for the satellite component class.
Code lines: 13
Contained by: module galacticus_nodes

function: satellitepositionattributematch
Description: Return a text list of component implementations in the satellite class that have the desired attributes for the position property
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: satellitepositionisgettable
Description: Returns true if the position property is gettable for the satellite component class.
Code lines: 7
19. Source Code Documentation

Contains by: module galacticus_nodes

function: satellitepresetboundmasshistoryget
Description: Return the boundMassHistory property of the SatellitePreset component implementation.
Code lines: 8
Contains by: module galacticus_nodes

subroutine: satellitepresetboundmasshistoryset
Description: Set the boundMassHistory property of the SatellitePreset component implementation.
Code lines: 9
Contains by: module galacticus_nodes
Modules used: memory_management

function: satellitepresetmergeboundmass
Description: Return the satellite bound mass at the current time.
Code lines: 32
Contains by: module galacticus_nodes
Modules used: memory_management

function: satellitepresetnodeindex
Description: Return the satellite node index.
Code lines: 35
Contains by: module galacticus_nodes
Modules used: memory_management

function: satellitepresetnodeindexhistoryget
Description: Return the nodeIndexHistory property of the SatellitePreset component implementation.
Code lines: 8
Contains by: module galacticus_nodes

subroutine: satellitepresetnodeindexhistoryset
Description: Set the nodeIndexHistory property of the SatellitePreset component implementation.
Code lines: 9
Contains by: module galacticus_nodes
Modules used: memory_management

function: satellitepresettimeofmergingget
Description: Return the timeOfMerging property of the SatellitePreset component implementation.
Code lines: 8
Contains by: module galacticus_nodes

subroutine: satellitepresettimeofmergingset
Description: Set the timeOfMerging property of the SatellitePreset component implementation.
Code lines: 9
function: satellitepresetvirialorbitget
Description: Return the virialOrbit property of the SatellitePreset component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satellitepresetvirialorbitset
Description: Set the virialOrbit property of the SatellitePreset component implementation.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

function: satellitestandardboundmasscount
Description: Return a count of the number of scalar properties in the boundMass property of the SatelliteStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: satellitestandardboundmassget
Description: Return the boundMass property of the SatelliteStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satellitestandardboundmassrate
Description: Accumulate to the boundMass property rate of change of the SatelliteStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: satellitestandardboundmassscale
Description: Set the boundMass property scale of the SatelliteStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satellitestandardboundmassset
Description: Set the boundMass property of the SatelliteStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: satellitestandardmergetimecount
Description: Return a count of the number of scalar properties in the mergeTime property of the SatelliteStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: satellitestandardmergetimerate
Description: Accumulate to the mergeTime property rate of change of the SatelliteStandard component implementation.
module galacticus_nodes

subroutine satellitestandardmergetimescale
Description: Set the mergeTime property scale of the SatelliteStandard component implementation.

subroutine satellitestandardmergetimeset
Description: Set the mergeTime property of the SatelliteStandard component implementation.

Modules used: memory_management

function satellitestandardvirialorbitget
Description: Get the value of the virialOrbit property of the SatelliteStandard component using a deferred function.

function satellitestandardvirialorbitgetfunction
Description: Set the function to be used for get of the virialOrbit property of the SatelliteStandard component class.

function satellitestandardvirialorbitgetisattached
Description: Return true if the deferred function used to get the virialOrbit property of the SatelliteStandard component class has been attached.

function satellitestandardvirialorbitgetvalue
Description: Return the virialOrbit property of the SatelliteStandard component implementation.

subroutine satellitestandardvirialorbitset
Description: Set the value of the virialOrbit property of the SatelliteStandard component using a deferred function.

subroutine satellitestandardvirialorbitsetfunction
Description: Set the function to be used for set of the virialOrbit property of the SatelliteStandard component class.

function satellitestandardvirialorbitsetisattached
19.1. Program units

Description: Return true if the deferred function used to set the virialOrbit property of the SatelliteStandard component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: satellitestandardvirialorbitsetvalue
Description: Set the virialOrbit property of the SatelliteStandard component implementation.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

function: satellitetidalheatingnormalized
Description: Returns the default value for the tidalHeatingNormalized property for the satellite component class.
Code lines: 8
Contained by: module galacticus_nodes

function: satellitetidalheatingnormalizedattributematch
Description: Return a text list of component implementations in the satellite class that have the desired attributes for the tidalHeatingNormalized property
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: satellitetidalheatingnormalizedisgettable
Description: Returns true if the tidalHeatingNormalized property is gettable for the satellite component class.
Code lines: 6
Contained by: module galacticus_nodes

function: satellitetidaltensorpathintegrated
Description: Returns the default value for the tidalTensorPathIntegrated property for the satellite component class.
Code lines: 8
Contained by: module galacticus_nodes

function: satellitetidaltensorpathintegratedattributematch
Description: Return a text list of component implementations in the satellite class that have the desired attributes for the tidalTensorPathIntegrated property
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: satellitetidaltensorpathintegratedisgettable
Description: Returns true if the tidalTensorPathIntegrated property is gettable for the satellite component class.
Code lines: 6
Contained by: module galacticus_nodes

function: satellitetimeofmerging
19. Source Code Documentation

**function**: satellitetimeofmergingattributematch

*Description:* Return a text list of component implementations in the satellite class that have the desired attributes for the timeOfMerging property.

*Code lines:* 24

*Contained by:* module galacticus_nodes

**function**: satellitetimeofmergingisgettable

*Description:* Returns true if the timeOfMerging property is gettable for the satellite component class.

*Code lines:* 60

*Contained by:* module galacticus_nodes

**function**: satellitevelocity

*Description:* Returns the default value for the velocity property for the satellite component class.

*Code lines:* 13

*Contained by:* module galacticus_nodes

**function**: satellitevelocityattributematch

*Description:* Return a text list of component implementations in the satellite class that have the desired attributes for the velocity property.

*Code lines:* 25

*Contained by:* module galacticus_nodes

**function**: satellitevelocityisgettable

*Description:* Returns true if the velocity property is gettable for the satellite component class.

*Code lines:* 10

*Contained by:* module galacticus_nodes

**function**: satelliteverysimplemergetimecount

*Description:* Return a count of the number of scalar properties in the mergeTime property of the SatelliteVerySimple component implementation.

*Code lines:* 1

*Contained by:* module galacticus_nodes

**subroutine**: satelliteverysimplemergetimerate

*Description:* Accumulate to the mergeTime property rate of change of the SatelliteVerySimple component implementation.

*Code lines:* 7

*Contained by:* module galacticus_nodes

**subroutine**: satelliteverysimplemergetimescale

*Description:* Set the mergeTime property scale of the SatelliteVerySimple component implementation.

*Code lines:* 8

*Contained by:* module galacticus_nodes
19.1. Program units

```
Contained by: module galacticus_nodes

subroutine: satelliteverysimplemergetimeset
Description: Set the mergeTime property of the SatelliteVerySimple component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: satellitevirialorbit
Description: Returns the default value for the virialOrbit property for the satellite component class.
Code lines: 8
Contained by: module galacticus_nodes

function: satellitevirialorbitattributematch
Description: Return a text list of component implementations in the satellite class that have the desired attributes for the virialOrbit property
Code lines: 49
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: satellitevirialorbitisgettable
Description: Returns true if the virialOrbit property is gettable for the satellite component class.
Code lines: 6
Contained by: module galacticus_nodes

function: serializetoarraycount
Description: Return a count of the size of the serialized treeNode object.
Code lines: 218
Contained by: module galacticus_nodes

subroutine: serializetoarrayrates
Description: Serialize rates to array.
Code lines: 307
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: serializetoarrayscales
Description: Serialize scales to array.
Code lines: 307
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: serializetoarrayvalues
Description: Serialize values to array.
Code lines: 307
Contained by: module galacticus_nodes
Modules used: memory_management

function: spheroidabundancesgas
```
Description: Returns the default value for the `abundancesGas` property for the `spheroid` component class.

Code lines: 8

Contained by: module `galacticus_nodes`

function: `spheroidabundancesgasattributematch`
Description: Return a text list of component implementations in the `spheroid` class that have the desired attributes for the `abundancesGas` property.

Code lines: 25

Contained by: module `galacticus_nodes`

Modules used: `iso_varying_string`

function: `spheroidabundancesgasisgettable`
Description: Returns true if the `abundancesGas` property is gettable for the `spheroid` component class.

Code lines: 6

Contained by: module `galacticus_nodes`

subroutine: `spheroidabundancesgasrate`
Description: Accept a rate set for the `abundancesGas` property of the `spheroid` component class. Trigger an interrupt to create the component.

Code lines: 15

Contained by: module `galacticus_nodes`

Modules used: `galacticus_error`

function: `spheroidabundancesstellar`
Description: Returns the default value for the `abundancesStellar` property for the `spheroid` component class.

Code lines: 8

Contained by: module `galacticus_nodes`

function: `spheroidabundancesstellarattributematch`
Description: Return a text list of component implementations in the `spheroid` class that have the desired attributes for the `abundancesStellar` property.

Code lines: 25

Contained by: module `galacticus_nodes`

Modules used: `iso_varying_string`

function: `spheroidabundancesstellarisgettable`
Description: Returns true if the `abundancesStellar` property is gettable for the `spheroid` component class.

Code lines: 6

Contained by: module `galacticus_nodes`

subroutine: `spheroidabundancesstellarrate`
Description: Accept a rate set for the `abundancesStellar` property of the `spheroid` component class. Trigger an interrupt to create the component.

Code lines: 15

Contained by: module `galacticus_nodes`

Modules used: `galacticus_error`

function: `spheroidangularmomentum`
19.1. Program units

**Description:** Returns the default value for the `angularMomentum` property for the `spheroid` component class.

**Code lines:** 8

**Contained by:** module `galacticus_nodes`

**function:** `spheroidangularmomentumattributematch`

**Description:** Return a text list of component implementations in the `spheroid` class that have the desired attributes for the `angularMomentum` property.

**Code lines:** 25

**Contained by:** module `galacticus_nodes`

**Modules used:** `iso_varying_string`

**function:** `spheroidangularmomentumisgettable`

**Description:** Returns true if the `angularMomentum` property is gettable for the `spheroid` component class.

**Code lines:** 6

**Contained by:** module `galacticus_nodes`

**subroutine:** `spheroidangularmomentumrate`

**Description:** Accept a rate set for the `angularMomentum` property of the `spheroid` component class. Trigger an interrupt to create the component.

**Code lines:** 15

**Contained by:** module `galacticus_nodes`

**Modules used:** `galacticus_error`

**function:** `spheroidcountlinked`

**Description:** Returns the number of `spheroid` components in `self`.

**Code lines:** 19

**Contained by:** module `galacticus_nodes`

**subroutine:** `spheroidcreatebyinterrupt`

**Description:** Create the `spheroid` component of `self` via an interrupt.

**Code lines:** 12

**Contained by:** module `galacticus_nodes`

**subroutine:** `spheroidcreatelinked`

**Description:** Create the `spheroid` component of `self`.

**Code lines:** 29

**Contained by:** module `galacticus_nodes`

**Modules used:** `galacticus_display` `iso_varying_string` `string_handling`

**subroutine:** `spheroiddestroylinked`

**Description:** Destroy the `spheroid` component of `self`.

**Code lines:** 13

**Contained by:** module `galacticus_nodes`

**function:** `spheroidenergygasinput`

**Description:** Returns the default value for the `energyGasInput` property for the `spheroid` component class.

**Code lines:** 8
19. Source Code Documentation

**function: spheroidenergygasinputattributematch**

*Description:* Return a text list of component implementations in the **spheroid** class that have the desired attributes for the **energyGasInput** property.

*Code lines:* 27

*Contained by:* module **galacticus_nodes**

*Modules used:* **iso_varying_string**

**function: spheroidenergygasinputisgettable**

*Description:* Returns true if the **energyGasInput** property is gettable for the **spheroid** component class.

*Code lines:* 6

*Contained by:* module **galacticus_nodes**

**function: spheroidget**

*Description:* Returns the **spheroid** component of **self**.

*Code lines:* 27

*Contained by:* module **galacticus_nodes**

*Modules used:* **galacticus_error**

**function: spheroidhalfmassradius**

*Description:* Returns the default value for the **halfMassRadius** property for the **spheroid** component class.

*Code lines:* 8

*Contained by:* module **galacticus_nodes**

**function: spheroidhalfmassradiusattributematch**

*Description:* Return a text list of component implementations in the **spheroid** class that have the desired attributes for the **halfMassRadius** property.

*Code lines:* 27

*Contained by:* module **galacticus_nodes**

*Modules used:* **iso_varying_string**

**function: spheroidhalfmassradiusisgettable**

*Description:* Returns true if the **halfMassRadius** property is gettable for the **spheroid** component class.

*Code lines:* 6

*Contained by:* module **galacticus_nodes**

**function: spheroidisinitialized**

*Description:* Returns the default value for the **isInitialized** property for the **spheroid** component class.

*Code lines:* 8

*Contained by:* module **galacticus_nodes**

**function: spheroidisinitializedattributematch**

*Description:* Return a text list of component implementations in the **spheroid** class that have the desired attributes for the **isInitialized** property.

*Code lines:* 27

*Contained by:* module **galacticus_nodes**

*Modules used:* **iso_varying_string**

**function: spheroidisinitializedisgettable**

*Description:* Returns true if the **isInitialized** property is gettable for the **spheroid** component class.

*Code lines:* 6

*Contained by:* module **galacticus_nodes**

1646
function: spheroidisinitializedisgettable
Description: Returns true if the isInitialized property is gettable for the spheroid component class.
Code lines: 6
Contained by: module galacticus_nodes

function: spheroidluminositiesstellar
Description: Returns the default value for the luminositiesStellar property for the spheroid component class.
Code lines: 8
Contained by: module galacticus_nodes

function: spheroidluminositiesstellarattributematch
Description: Return a text list of component implementations in the spheroid class that have the desired attributes for the luminositiesStellar property
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: spheroidluminositiesstellarisgettable
Description: Returns true if the luminositiesStellar property is gettable for the spheroid component class.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: spheroidluminositiesstellarrate
Description: Accept a rate set for the luminositiesStellar property of the spheroid component class. Trigger an interrupt to create the component.
Code lines: 15
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: spheroidmassgas
Description: Returns the default value for the massGas property for the spheroid component class.
Code lines: 8
Contained by: module galacticus_nodes

function: spheroidmassgasattributematch
Description: Return a text list of component implementations in the spheroid class that have the desired attributes for the massGas property
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: spheroidmassgasisgettable
Description: Returns true if the massGas property is gettable for the spheroid component class.
Code lines: 6
Contained by: module galacticus_nodes
19. Source Code Documentation

subroutine: spheroidmassgasrate
Description: Accept a rate set for the massGas property of the spheroid component class. Trigger an interrupt to create the component.
Code lines: 15
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: spheroidmassgassink
Description: Returns the default value for the massGasSink property for the spheroid component class.
Code lines: 8
Contained by: module galacticus_nodes

function: spheroidmassgassinkattributematch
Description: Return a text list of component implementations in the spheroid class that have the desired attributes for the massGasSink property.
Code lines: 27
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: spheroidmassgassinkisgettable
Description: Returns true if the massGasSink property is gettable for the spheroid component class.
Code lines: 6
Contained by: module galacticus_nodes

function: spheroidmassstellar
Description: Returns the default value for the massStellar property for the spheroid component class.
Code lines: 8
Contained by: module galacticus_nodes

function: spheroidmassstellarattributematch
Description: Return a text list of component implementations in the spheroid class that have the desired attributes for the massStellar property.
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: spheroidmassstellarisgettable
Description: Returns true if the massStellar property is gettable for the spheroid component class.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: spheroidmassstellarrate
Description: Accept a rate set for the massStellar property of the spheroid component class. Trigger an interrupt to create the component.
Code lines: 15
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: spheroidnullbindingabundances0inout
19.1. Program units

- **Description:** A null get function for rank 0 type abundances.
  
  **Code lines:** 7

  **Contained by:** module `galacticus_nodes`

- **function:** `spheroidnullbindingdouble0inout`

  **Description:** A null get function for rank 0 double precisions.

  **Code lines:** 7

  **Contained by:** module `galacticus_nodes`

- **function:** `spheroidnullbindinghistory0inout`

  **Description:** A null get function for rank 0 type histories.

  **Code lines:** 7

  **Contained by:** module `galacticus_nodes`

- **function:** `spheroidnullbindinginteger0in`

  **Description:** A null get function for rank 0 integers.

  **Code lines:** 7

  **Contained by:** module `galacticus_nodes`

- **function:** `spheroidnullbindinglogical0inout`

  **Description:** A null get function for rank 0 logicals.

  **Code lines:** 7

  **Contained by:** module `galacticus_nodes`

- **subroutine:** `spheroidnullbindingrateabundances0inout`

  **Description:** A null rate function for rank 0 type abundances.

  **Code lines:** 9

  **Contained by:** module `galacticus_nodes`

- **subroutine:** `spheroidnullbindingratedouble0inout`

  **Description:** A null rate function for rank 0 double precisions.

  **Code lines:** 9

  **Contained by:** module `galacticus_nodes`

- **subroutine:** `spheroidnullbindingratehistory0inout`

  **Description:** A null rate function for rank 0 type histories.

  **Code lines:** 9

  **Contained by:** module `galacticus_nodes`

- **subroutine:** `spheroidnullbindingrateinteger0in`

  **Description:** A null rate function for rank 0 integers.

  **Code lines:** 9

  **Contained by:** module `galacticus_nodes`

- **subroutine:** `spheroidnullbindingratelogical0inout`

  **Description:** A null rate function for rank 0 logicals.

  **Code lines:** 9

  **Contained by:** module `galacticus_nodes`
19. Source Code Documentation

subroutine: spheroidnullbindingratestellarluminosities0inout
Description: A null rate function for rank 0 type stellar luminosities.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: spheroidnullbindingsetabundances0inout
Description: A null set function for rank 0 type abundances.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: spheroidnullbindingsetdouble0inout
Description: A null set function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: spheroidnullbindingsethistory0inout
Description: A null set function for rank 0 type histories.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: spheroidnullbindingsetinteger0in
Description: A null set function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: spheroidnullbindingsetlogical0inout
Description: A null set function for rank 0 logicals.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: spheroidnullbindingsetstellarluminosities0inout
Description: A null set function for rank 0 type stellar luminosities.
Code lines: 7
Contained by: module galacticus_nodes

function: spheroidnullbindingstellarluminosities0inout
Description: A null get function for rank 0 type stellar luminosities.
Code lines: 7
Contained by: module galacticus_nodes

function: spheroidradius
Description: Returns the default value for the radius property for the spheroid component class.
Code lines: 8
Contained by: module galacticus_nodes

function: spheroidradiusattributematch
Description: Return a text list of component implementations in the spheroid class that have the desired attributes for the radius property.
Code lines: 27
19.1. Program units

function: spheroidradiusisgettable
Description: Returns true if the radius property is gettable for the spheroid component class.
Code lines: 6
Contained by: module galacticus_nodes

function: spheroidstandardabundancesgascount
Description: Return a count of the number of scalar properties in the abundancesGas property of the spheroidStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: spheroidstandardabundancesgasget
Description: Return the abundancesGas property of the SpheroidStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: spheroidstandardabundancesgasrate
Description: Accumulate to the abundancesGas property rate of change of the SpheroidStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: spheroidstandardabundancesgasrategeneric
Description: Set the rate of the abundancesGas property of the SpheroidStandard component via a generic nodeComponent.
Code lines: 24
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: spheroidstandardabundancesgasscale
Description: Set the abundancesGas property scale of the SpheroidStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: spheroidstandardabundancesgasset
Description: Set the abundancesGas property of the SpheroidStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: spheroidstandardabundancesstellarcount
Description: Return a count of the number of scalar properties in the abundancesStellar property of the spheroidStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes
function: spheroidstandardabundancesstellarget
Description: Return the abundancesStellar property of the SpheroidStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: spheroidstandardabundancesstellarrate
Description: Accumulate to the abundancesStellar property rate of change of the SpheroidStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: spheroidstandardabundancesstellarrategeneric
Description: Set the rate of the abundancesStellar property of the SpheroidStandard component via a generic nodeComponent.
Code lines: 24
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: spheroidstandardabundancesstellarscale
Description: Set the abundancesStellar property scale of the SpheroidStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: spheroidstandardabundancesstellarset
Description: Set the abundancesStellar property of the SpheroidStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: spheroidstandardangularmomentumcount
Description: Return a count of the number of scalar properties in the angularMomentum property of the spheroidStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: spheroidstandardangularmomentumget
Description: Return the angularMomentum property of the SpheroidStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: spheroidstandardangularmomentumrate
Description: Accumulate to the angularMomentum property rate of change of the SpheroidStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: spheroidstandardangularmomentumrategeneric
19.1. Program units

*Description:* Set the rate of the \texttt{angularMomentum} property of the \texttt{SpheroidStandard} component via a generic \texttt{nodeComponent}.

*Code lines:* 24

*Contained by:* module \texttt{galacticus_nodes}

*Modules used:* \texttt{galacticus_error}

**subroutine:** \texttt{spheroidstandardangularmomentumscale}

*Description:* Set the \texttt{angularMomentum} property scale of the \texttt{SpheroidStandard} component implementation.

*Code lines:* 8

*Contained by:* module \texttt{galacticus_nodes}

**subroutine:** \texttt{spheroidstandardangularmomentumset}

*Description:* Set the \texttt{angularMomentum} property of the \texttt{SpheroidStandard} component implementation.

*Code lines:* 11

*Contained by:* module \texttt{galacticus_nodes}

*Modules used:* \texttt{memory_management}

**subroutine:** \texttt{spheroidstandardcreatefunctionset}

*Description:* Set the create function for the \texttt{SpheroidStandard} component.

*Code lines:* 6

*Contained by:* module \texttt{galacticus_nodes}

**subroutine:** \texttt{spheroidstandardenergygasinputrate}

*Description:* Set the rate of the \texttt{energyGasInput} property of the \texttt{SpheroidStandard} component using a deferred function.

*Code lines:* 10

*Contained by:* module \texttt{galacticus_nodes}

**subroutine:** \texttt{spheroidstandardenergygasinputratefunction}

*Description:* Set the function to be used for rate of the \texttt{energyGasInput} property of the \texttt{SpheroidStandard} component class.

*Code lines:* 8

*Contained by:* module \texttt{galacticus_nodes}

**function:** \texttt{spheroidstandardenergygasinputrateisattached}

*Description:* Return true if the deferred function used to rate the \texttt{energyGasInput} property of the \texttt{SpheroidStandard} component class has been attached.

*Code lines:* 5

*Contained by:* module \texttt{galacticus_nodes}

**function:** \texttt{spheroidstandardisinitializedget}

*Description:* Return the \texttt{is Initialized} property of the \texttt{SpheroidStandard} component implementation.

*Code lines:* 8

*Contained by:* module \texttt{galacticus_nodes}

**subroutine:** \texttt{spheroidstandardisinitializedset}

*Description:* Set the \texttt{is Initialized} property of the \texttt{SpheroidStandard} component implementation.

*Code lines:* 9

*Contained by:* module \texttt{galacticus_nodes}
19. Source Code Documentation

Modules used: memory_management

function: spheroidstandardluminositiesstellarcoun
Description: Return a count of the number of scalar properties in the luminositiesStellar property of the spheroidStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: spheroidstandardluminositiesstellarget
Description: Return the luminositiesStellar property of the SpheroidStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: spheroidstandardluminositiesstellarrate
Description: Accumulate to the luminositiesStellar property rate of change of the SpheroidStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: spheroidstandardluminositiesstellarrategeneric
Description: Set the rate of the luminositiesStellar property of the SpheroidStandard component via a generic nodeComponent.
Code lines: 24
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: spheroidstandardluminositiesstellarscale
Description: Set the luminositiesStellar property scale of the SpheroidStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: spheroidstandardluminositiesstellarset
Description: Set the luminositiesStellar property of the SpheroidStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: spheroidstandardmassgascount
Description: Return a count of the number of scalar properties in the massGas property of the spheroidStandard component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: spheroidstandardmassgasget
Description: Return the massGas property of the SpheroidStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes
**19.1. Program units**

**subroutine: spheroidstandardmassgasrate**
*Description:* Accumulate to the `massGas` property rate of change of the `SpheroidStandard` component implementation.
*Code lines:* 10
*Contained by:* module `galacticus_nodes`

**subroutine: spheroidstandardmassgasrategeneric**
*Description:* Set the rate of the `massGas` property of the `SpheroidStandard` component via a generic `nodeComponent`.
*Code lines:* 24
*Contained by:* module `galacticus_nodes`
*Modules used:* `galacticus_error`

**subroutine: spheroidstandardmassgasscale**
*Description:* Set the `massGas` property scale of the `SpheroidStandard` component implementation.
*Code lines:* 8
*Contained by:* module `galacticus_nodes`

**subroutine: spheroidstandardmassgasset**
*Description:* Set the `massGas` property of the `SpheroidStandard` component implementation.
*Code lines:* 11
*Contained by:* module `galacticus_nodes`
*Modules used:* `memory_management`

**subroutine: spheroidstandardmassgassinkrate**
*Description:* Set the rate of the `massGasSink` property of the `SpheroidStandard` component using a deferred function.
*Code lines:* 10
*Contained by:* module `galacticus_nodes`

**subroutine: spheroidstandardmassgassinkratefunction**
*Description:* Set the function to be used for rate of the `massGasSink` property of the `SpheroidStandard` component class.
*Code lines:* 8
*Contained by:* module `galacticus_nodes`

**function: spheroidstandardmassgassinkrateisattached**
*Description:* Return true if the deferred function used to rate the `massGasSink` property of the `SpheroidStandard` component class has been attached.
*Code lines:* 5
*Contained by:* module `galacticus_nodes`

**function: spheroidstandardmassstellarcount**
*Description:* Return a count of the number of scalar properties in the `massStellar` property of the `spheroidStandard` component implementation.
*Code lines:* 7
*Contained by:* module `galacticus_nodes`

**function: spheroidstandardmassstellarget**
**Description:** Return the massStellar property of the SpheroidStandard component implementation.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**subroutine: spheroidstandardmassstellarrate**

**Description:** Accumulate to the massStellar property rate of change of the SpheroidStandard component implementation.

**Code lines:** 10

**Contained by:** module galacticus_nodes

**subroutine: spheroidstandardmassstellarrategeneric**

**Description:** Set the rate of the massStellar property of the SpheroidStandard component via a generic nodeComponent.

**Code lines:** 24

**Contained by:** module galacticus_nodes

**Modules used:** galacticus_error

**subroutine: spheroidstandardmassstellarscale**

**Description:** Set the massStellar property scale of the SpheroidStandard component implementation.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**subroutine: spheroidstandardmassstellarset**

**Description:** Set the massStellar property of the SpheroidStandard component implementation.

**Code lines:** 11

**Contained by:** module galacticus_nodes

**Modules used:** memory_management

**function: spheroidstandardradiusget**

**Description:** Return the radius property of the SpheroidStandard component implementation.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**subroutine: spheroidstandardradiusset**

**Description:** Set the radius property of the SpheroidStandard component implementation.

**Code lines:** 9

**Contained by:** module galacticus_nodes

**Modules used:** memory_management

**function: spheroidstandardstarformationhistorycount**

**Description:** Return a count of the number of scalar properties in the starFormationHistory property of the spheroidStandard component implementation.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**function: spheroidstandardstarformationhistoryget**

**Description:** Return the starFormationHistory property of the SpheroidStandard component implementation.

**Code lines:** 8

**Contained by:** module galacticus_nodes
**19.1. Program units**

**subroutine: spheroidstandardstarformationhistoryrate**
*Description:* Accumulate to the starFormationHistory property rate of change of the SpheroidStandard component implementation.
*Code lines:* 10
*Contained by:* module galacticus_nodes

**subroutine: spheroidstandardstarformationhistoryrategeneric**
*Description:* Set the rate of the starFormationHistory property of the SpheroidStandard component via a generic nodeComponent.
*Code lines:* 24
*Contained by:* module galacticus_nodes
*Modules used:* galacticus_error

**subroutine: spheroidstandardstarformationhistoryscale**
*Description:* Set the starFormationHistory property scale of the SpheroidStandard component implementation.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine: spheroidstandardstarformationhistoryset**
*Description:* Set the starFormationHistory property of the SpheroidStandard component implementation.
*Code lines:* 11
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**function: spheroidstandardstarformationrateget**
*Description:* Get the value of the starFormationRate property of the SpheroidStandard component using a deferred function.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine: spheroidstandardstarformationrategetfunction**
*Description:* Set the function to be used for get of the starFormationRate property of the SpheroidStandard component class.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**function: spheroidstandardstarformationrategetisattached**
*Description:* Return true if the deferred function used to get the starFormationRate property of the SpheroidStandard component class has been attached.
*Code lines:* 5
*Contained by:* module galacticus_nodes

**function: spheroidstandardstellarpropertieshistorycount**
*Description:* Return a count of the number of scalar properties in the stellarPropertiesHistory property of the spheroidStandard component implementation.
*Code lines:* 7
*Contained by:* module galacticus_nodes
function: spheroidstandardstellarpropertieshistoryget
Description: Return the stellarPropertiesHistory property of the SpheroidStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: spheroidstandardstellarpropertieshistoryrate
Description: Accumulate to the stellarPropertiesHistory property rate of change of the SpheroidStandard component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: spheroidstandardstellarpropertieshistoryrategeneric
Description: Set the rate of the stellarPropertiesHistory property of the SpheroidStandard component via a generic nodeComponent.
Code lines: 24
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: spheroidstandardstellarpropertieshistoryscale
Description: Set the stellarPropertiesHistory property scale of the SpheroidStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: spheroidstandardstellarpropertieshistoryset
Description: Set the stellarPropertiesHistory property of the SpheroidStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes
Modules used: memory_management

function: spheroidstandardvelocityget
Description: Return the velocity property of the SpheroidStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: spheroidstandardvelocityset
Description: Set the velocity property of the SpheroidStandard component implementation.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

function: spheroidstarformationhistory
Description: Returns the default value for the starFormationHistory property for the spheroid component class.
Code lines: 8
Contained by: module galacticus_nodes
19.1. Program units

function: spheroidstarformationhistoryattributematch
Description: Return a text list of component implementations in the spheroid class that have the desired attributes for the starFormationHistory property
Code lines: 25
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: spheroidstarformationhistoryisgettable
Description: Returns true if the starFormationHistory property is gettable for the spheroid component class.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: spheroidstarformationhistoryrate
Description: Accept a rate set for the starFormationHistory property of the spheroid component class. Trigger an interrupt to create the component.
Code lines: 15
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: spheroidstarformationrate
Description: Returns the default value for the starFormationRate property for the spheroid component class.
Code lines: 8
Contained by: module galacticus_nodes

function: spheroidstarformationrateattributematch
Description: Return a text list of component implementations in the spheroid class that have the desired attributes for the starFormationRate property
Code lines: 27
Contained by: module galacticus_nodes
Modules used: iso_varying_string

function: spheroidstarformationrateisgettable
Description: Returns true if the starFormationRate property is gettable for the spheroid component class.
Code lines: 6
Contained by: module galacticus_nodes

function: spheroidstellarpropertieshistory
Description: Returns the default value for the stellarPropertiesHistory property for the spheroid component class.
Code lines: 8
Contained by: module galacticus_nodes

function: spheroidstellarpropertieshistoryattributematch
Description: Return a text list of component implementations in the spheroid class that have the desired attributes for the stellarPropertiesHistory property
Code lines: 25
Contained by: module galacticus_nodes
19. Source Code Documentation

**Modules used:** iso_varying_string

**function:** spheroidstellarpropertieshistoryisgettable

*Description:* Returns true if the stellarPropertiesHistory property is gettable for the spheroid component class.

*Code lines:* 6

*Contained by:* module galacticus_nodes

**subroutine:** spheroidstellarpropertieshistoryrate

*Description:* Accept a rate set for the stellarPropertiesHistory property of the spheroid component class. Trigger an interrupt to create the component.

*Code lines:* 15

*Contained by:* module galacticus_nodes

**Modules used:** galacticus_error

**function:** spheroidvelocity

*Description:* Returns the default value for the velocity property for the spheroid component class.

*Code lines:* 8

*Contained by:* module galacticus_nodes

**function:** spheroidvelocityattributematch

*Description:* Return a text list of component implementations in the spheroid class that have the desired attributes for the velocity property

*Code lines:* 27

*Contained by:* module galacticus_nodes

**Modules used:** iso_varying_string

**function:** spheroidvelocityisgettable

*Description:* Returns true if the velocity property is gettable for the spheroid component class.

*Code lines:* 6

*Contained by:* module galacticus_nodes

**function:** spincountlinked

*Description:* Returns the number of spin components in self.

*Code lines:* 19

*Contained by:* module galacticus_nodes

**subroutine:** spincreatebyinterrupt

*Description:* Create the spin component of self via an interrupt.

*Code lines:* 8

*Contained by:* module galacticus_nodes

**subroutine:** spincreatelinked

*Description:* Create the spin component of self.

*Code lines:* 29

*Contained by:* module galacticus_nodes

*Modules used:* galacticus_display iso_varying_string string_handling
subroutine: spindestroylinked
Description: Destroy the spin component of self.
Code lines: 13
Contained by: module galacticus_nodes

function: spinget
Description: Returns the spin component of self.
Code lines: 27
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: spinnullbindingdouble0inout
Description: A null get function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

function: spinnullbindingdouble1inout
Description: A null get function for rank 1 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

function: spinnullbindinginteger0in
Description: A null get function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: spinnullbindingratedouble0inout
Description: A null rate function for rank 0 double precisions.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: spinnullbindingratedouble1inout
Description: A null rate function for rank 1 double precisions.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: spinnullbindingrateinteger0in
Description: A null rate function for rank 0 integers.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: spinnullbindingsetdouble0inout
Description: A null set function for rank 0 double precisions.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: spinnullbindingsetdouble1inout
Description: A null set function for rank 1 double precisions.
subroutine: spinnullbindingsetinteger0in
Description: A null set function for rank 0 integers.
Code lines: 7
Contained by: module galacticus_nodes

function: spinpreset3dspinvectorcount
Description: Return a count of the number of scalar properties in the spinVector property of the spinPreset3D component implementation.
Code lines: 11
Contained by: module galacticus_nodes

function: spinpreset3dspinvectorget
Description: Return the spinVector property of the SpinPreset3D component implementation.
Code lines: 8
Contained by: module galacticus_nodes

function: spinpreset3dspinvectorgrowthrateget
Description: Return the spinVectorGrowthRate property of the SpinPreset3D component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: spinpreset3dspinvectorgrowthrateset
Description: Set the spinVectorGrowthRate property of the SpinPreset3D component implementation.
Code lines: 17
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: spinpreset3dspinvectorrate
Description: Accumulate to the spinVector property rate of change of the SpinPreset3D component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: spinpreset3dspinvectorscale
Description: Set the spinVector property scale of the SpinPreset3D component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: spinpreset3dspinvectorset
Description: Set the spinVector property of the SpinPreset3D component implementation.
Code lines: 35
Contained by: module galacticus_nodes
Modules used: memory_management

function: spinpresetspincount
<table>
<thead>
<tr>
<th>Description</th>
<th>Code lines</th>
<th>Contained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return a count of the number of scalar properties in the spin property of the spinPreset component implementation.</td>
<td>7</td>
<td>module galacticus_nodes</td>
</tr>
</tbody>
</table>

**function: spinpresetspinget**

<table>
<thead>
<tr>
<th>Description</th>
<th>Code lines</th>
<th>Contained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return the spin property of the SpinPreset component implementation.</td>
<td>8</td>
<td>module galacticus_nodes</td>
</tr>
</tbody>
</table>

**function: spinpresetspingrowthrateget**

<table>
<thead>
<tr>
<th>Description</th>
<th>Code lines</th>
<th>Contained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return the spinGrowthRate property of the SpinPreset component implementation.</td>
<td>8</td>
<td>module galacticus_nodes</td>
</tr>
</tbody>
</table>

**subroutine: spinpresetspingrowthrateset**

<table>
<thead>
<tr>
<th>Description</th>
<th>Code lines</th>
<th>Contained by</th>
<th>Modules used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set the spinGrowthRate property of the SpinPreset component implementation.</td>
<td>9</td>
<td>module galacticus_nodes</td>
<td>memory_management</td>
</tr>
</tbody>
</table>

**subroutine: spinpresetspinrate**

<table>
<thead>
<tr>
<th>Description</th>
<th>Code lines</th>
<th>Contained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accumulate to the spin property rate of change of the SpinPreset component implementation.</td>
<td>10</td>
<td>module galacticus_nodes</td>
</tr>
</tbody>
</table>

**subroutine: spinpresetspinscale**

<table>
<thead>
<tr>
<th>Description</th>
<th>Code lines</th>
<th>Contained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set the spin property scale of the SpinPreset component implementation.</td>
<td>8</td>
<td>module galacticus_nodes</td>
</tr>
</tbody>
</table>

**subroutine: spinpresetspinset**

<table>
<thead>
<tr>
<th>Description</th>
<th>Code lines</th>
<th>Contained by</th>
<th>Modules used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set the spin property of the SpinPreset component implementation.</td>
<td>11</td>
<td>module galacticus_nodes</td>
<td>memory_management</td>
</tr>
</tbody>
</table>

**function: spinrandomspincount**

<table>
<thead>
<tr>
<th>Description</th>
<th>Code lines</th>
<th>Contained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return a count of the number of scalar properties in the spin property of the spinRandom component implementation.</td>
<td>7</td>
<td>module galacticus_nodes</td>
</tr>
</tbody>
</table>

**function: spinrandomspinget**

<table>
<thead>
<tr>
<th>Description</th>
<th>Code lines</th>
<th>Contained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return the spin property of the SpinRandom component implementation.</td>
<td>8</td>
<td>module galacticus_nodes</td>
</tr>
</tbody>
</table>
**function: spinrandomspingrowthrate**
*Description:* Return the spinGrowthRate property of the SpinRandom component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine: spinrandomspinrate**
*Description:* Accumulate to the spin property rate of change of the SpinRandom component implementation.
*Code lines:* 10
*Contained by:* module galacticus_nodes

**subroutine: spinrandomspinscale**
*Description:* Set the spin property scale of the SpinRandom component implementation.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**subroutine: spinrandomspinset**
*Description:* Set the spin property of the SpinRandom component implementation.
*Code lines:* 11
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**function: spinspin**
*Description:* Returns the default value for the spin property for the spin component class.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**function: spinspinattributematch**
*Description:* Return a text list of component implementations in the spin class that have the desired attributes for the spin property.
*Code lines:* 34
*Contained by:* module galacticus_nodes
*Modules used:* iso_varying_string

**function: spinspingrowthrate**
*Description:* Returns the default value for the spinGrowthRate property for the spin component class.
*Code lines:* 8
*Contained by:* module galacticus_nodes

**function: spinspingrowthrateattributematch**
*Description:* Return a text list of component implementations in the spin class that have the desired attributes for the spinGrowthRate property.
*Code lines:* 38
*Contained by:* module galacticus_nodes
*Modules used:* iso_varying_string

**function: spinspingrowthrateisgettable**
*Description:* Returns true if the spinGrowthRate property is gettable for the spin component class.
**19.1. Program units**

<table>
<thead>
<tr>
<th>Code lines</th>
<th>Description</th>
<th>Contained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Returns true if the <code>spin</code> property is gettable for the <code>spin</code> component class.</td>
<td>module <code>galacticus_nodes</code></td>
</tr>
</tbody>
</table>

**function: spinspinisgettable**

<table>
<thead>
<tr>
<th>Code lines</th>
<th>Description</th>
<th>Contained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Returns the default value for the <code>spinVector</code> property for the <code>spin</code> component class.</td>
<td>module <code>galacticus_nodes</code></td>
</tr>
</tbody>
</table>

**function: spinspinvector**

<table>
<thead>
<tr>
<th>Code lines</th>
<th>Description</th>
<th>Contained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>Return a text list of component implementations in the <code>spin</code> class that have the desired attributes for the <code>spinVector</code> property.</td>
<td>module <code>galacticus_nodes</code></td>
</tr>
</tbody>
</table>

**function: spinspinvectorattributematch**

<table>
<thead>
<tr>
<th>Code lines</th>
<th>Description</th>
<th>Contained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>Returns true if the <code>spinVectorGrowthRate</code> property is gettable for the <code>spin</code> component class.</td>
<td>module <code>galacticus_nodes</code></td>
</tr>
</tbody>
</table>

**function: spinspinvectorgrowthrate**

<table>
<thead>
<tr>
<th>Code lines</th>
<th>Description</th>
<th>Contained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>Returns true if the <code>spinVectorGrowthRate</code> property is gettable for the <code>spin</code> component class.</td>
<td>module <code>galacticus_nodes</code></td>
</tr>
</tbody>
</table>

**function: tree_node_component_builder**

<table>
<thead>
<tr>
<th>Code lines</th>
<th>Description</th>
<th>Contained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>362</td>
<td>Build components in a <code>treeNode</code> object given an XML definition.</td>
<td>module <code>galacticus_nodes</code></td>
</tr>
</tbody>
</table>

**subroutine: tree_node_component_builder**

<table>
<thead>
<tr>
<th>Code lines</th>
<th>Description</th>
<th>Contained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>362</td>
<td>Build components in a <code>treeNode</code> object given an XML definition.</td>
<td>module <code>galacticus_nodes</code></td>
</tr>
</tbody>
</table>

** Modules used:**

- `iso_varying_string`
function: tree_node_constructor
Description: Return a pointer to a newly created and initialized treeNode.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: tree_node_copy_node_to
Description: Make a copy of self in targetNode.
Code lines: 510
Contained by: module galacticus_nodes

function: tree_node_create_event
Description: Create a new event in a tree node.
Code lines: 21
Contained by: module galacticus_nodes

subroutine: tree_node_create_initialize
Description: Initializes tree node create by calling all relevant initialization routines.
Code lines: 15
Contained by: module galacticus_nodes
Modules used: input_parameters

function: tree_node_get_earliest_progenitor
Description: Returns a pointer to the earliest progenitor of self.
Code lines: 14
Contained by: module galacticus_nodes

function: tree_node_get_last_satellite
Description: Returns a pointer to the final satellite node associated with self.
Code lines: 11
Contained by: module galacticus_nodes

function: tree_node_index
Description: Returns the index of a treeNode.
Code lines: 17
Contained by: module galacticus_nodes

subroutine: tree_node_index_set
Description: Sets the index of a treeNode.
Code lines: 8
Contained by: module galacticus_nodes

function: tree_node_is_on_main_branch
Description: Returns true if self is on the main branch.
Code lines: 17
Contained by: module galacticus_nodes

function: tree_node_is_primary_progenitor
Description: Returns true if self is the primary progenitor of its parent node.
Code lines: 14
Contained by: module galacticus_nodes

function: tree_node_is_primary_progenitor_of_index
Description: Return true if self is a progenitor of the node with index targetNodeIndex.
Code lines: 21
Contained by: module galacticus_nodes

function: tree_node_is_primary_progenitor_of_node
Description: Return true if self is a progenitor of targetNode.
Code lines: 21
Contained by: module galacticus_nodes

function: tree_node_is_progenitor_of_index
Description: Return true if self is a progenitor of the node with index targetNodeIndex.
Code lines: 20
Contained by: module galacticus_nodes

function: tree_node_is_progenitor_of_node
Description: Return true if self is a progenitor of targetNode.
Code lines: 20
Contained by: module galacticus_nodes

function: tree_node_is_satellite
Description: Returns true if self is a satellite.
Code lines: 27
Contained by: module galacticus_nodes

function: tree_node_merges_with_node
Description: Returns a pointer to the node with which thisNode will merge.
Code lines: 15
Contained by: module galacticus_nodes

subroutine: tree_node_move_components
Description: Move components from self to targetNode.
Code lines: 224
Contained by: module galacticus_nodes

subroutine: tree_node_ode_step_rates_initialize
Description: Initialize the rates in components of tree node self in preparation for an ODE solver step.
Code lines: 217
Contained by: module galacticus_nodes

subroutine: tree_node_ode_step_scales_initialize
Description: Initialize the scales in components of tree node self in preparation for an ODE solver step.
Code lines: 217
Contained by: module galacticus_nodes

subroutine: tree_node_remove_from_host
Description: Remove self from the linked list of its host node's satellites.
Code lines: 36
Contained by: module galacticus_nodes
Modules used: galacticus_display string_handling

subroutine: tree_node_remove_from_mergee
Description: Remove self from the linked list of its host node's satellites.
Code lines: 38
Contained by: module galacticus_nodes
Modules used: galacticus_display string_handling

subroutine: tree_node_remove_paired_event
Description: Removed a paired event from self. Matching is done on the basis of event ID.
Code lines: 33
Contained by: module galacticus_nodes

function: tree_node_type
Description: Returns the name of a treeNode object.
Code lines: 8
Contained by: module galacticus_nodes

function: tree_node_unique_id
Description: Returns the unique ID of a treeNode.
Code lines: 17
Contained by: module galacticus_nodes

subroutine: tree_node_unique_id_set
Description: Sets the index of a treeNode.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: tree_node_walk_branch
Description: This function provides a mechanism for walking through the branches of the merger tree. Given a pointer self to a branch of the tree, it will return the next node that should be visited in the tree. Thus, if self is initially set to the base of the merger tree and Merger._Tree_Walk_Branch() is called repeatedly it will walk through every node of the branch. Once the entire branch has been walked, a null() pointer will be returned, indicating that there are no more nodes to walk. Each node will be visited once and once only if the branch is walked in this way.
Code lines: 35
Contained by: module galacticus_nodes

subroutine: tree_node_walk_branch_with_satellites


Description: This function provides a mechanism for walking through the branches of the merger tree. Given a pointer self to a branch of the tree, it will return the next node that should be visited in the tree. Thus, if self is initially set to the base of the merger tree and Merger_Tree_Walk_Branch() is called repeatedly it will walk through every node of the branch. Once the entire branch has been walked, a null() pointer will be returned, indicating that there are no more nodes to walk. Each node will be visited once and once only if the branch is walked in this way. Note that it is important that the walk descends to satellites before descending to children: the routines that destroy merger tree branches rely on this since child nodes are used in testing whether a node is a satellite—if they are destroyed prior to the test being made then problems with dangling pointers will occur.

Code lines: 53
Contained by: module galacticus_nodes

subroutine: tree_node_walk_tree
**Description:** This function provides a mechanism for walking through an entire merger tree. Given a pointer `self` to a node of the tree, it will return the next node that should be visited in the tree. Thus, if `self` is initially set to the base of the merger tree and `Merger_Tree_Walk()` is called repeatedly it will walk through every node of the tree. Once the entire tree has been walked, a `null()` pointer will be returned, indicating that there are no more nodes to walk. Each node will be visited once and once only if the tree is walked in this way.

**Code lines:** 36  
**Contained by:** module galacticus_nodes

**subroutine:** tree_node_walk_tree_under_construction  
**Description:** This function provides a mechanism for walking through a merger tree that is being built.  
**Code lines:** 42  
**Contained by:** module galacticus_nodes

**subroutine:** tree_node_walk_tree_with_satellites  
**Description:** Merger tree walk function which also descends through satellite nodes. Note that it is important that the walk descends to satellites before descending to children: the routines that destroy merger tree branches rely on this since child nodes are used in testing whether a node is a satellite—if they are destroyed prior to the test being made then problems with dangling pointers will occur.  
**Code lines:** 48  
**Contained by:** module galacticus_nodes

**type:** treeevent  
**Description:** Type for events attached to trees.  
**Code lines:** 8  
**Contained by:** module galacticus_nodes

**type:** treenode  
**Description:** A class for nodes in merger trees.  
**Code lines:** 1112  
**Contained by:** module galacticus_nodes

**subroutine:** treenodedestroy  
**Description:** Destroy a `treeNode` object.  
**Code lines:** 57  
**Contained by:** module galacticus_nodes

**subroutine:** treenodeinitialize  
**Description:** Initialize a `treeNode` object.  
**Code lines:** 68  
**Contained by:** module galacticus_nodes

**Modules used:** galacticus_error

**type:** treenodelist  
**Description:** Type to give a list of treeNodes.  
**Code lines:** 3  
**Contained by:** module galacticus_nodes
type: universe
Description: The universe object class.
Code lines: 37
Contained by: module galacticus_nodes

function: universecreateevent
Description: Create a new event in a universe.
Code lines: 21
Contained by: module galacticus_nodes

type: universeevent
Description: Type for events attached to universes.
Code lines: 8
Contained by: module galacticus_nodes

function: universepoptree
Description: Pop a tree from the universe.
Code lines: 16
Contained by: module galacticus_nodes

subroutine: universepushtree
Description: Pop a tree from the universe.
Code lines: 20
Contained by: module galacticus_nodes

subroutine: universeremoveevent
Description: Remove an event from self.
Code lines: 27
Contained by: module galacticus_nodes

file: objects.nodes.bindings.C.F90
Code lines: 943

function: cnode_component_agestatistics_diskintegratedsfr
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes iso_c_binding

function: cnode_component_agestatistics_disktimeweightedintegratedsfr
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes iso_c_binding

function: cnode_component_agestatistics_spheroidintegratedsfr
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes iso_c_binding
function: cnode_component_agestatistics_spheroidtimeweightedintegratedsfr
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_basic_accretionrate
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_basic_mass
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_basic_massmaximum
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_basic_time
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_basic_timelastisolated
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_blackhole_accretionrate
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_blackhole_mass
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_blackhole_massseed
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_blackhole_radialposition
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90

1672
19.1. Program units

Modules used: galacticus_nodes iso_c_binding

function: cnode_component_blackhole_radiativeefficiency
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_blackhole_spin
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_blackhole_spinseed
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_blackhole_tripleinteractiontime
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_darkmatterprofile_scale
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_darkmatterprofile_scalegrowthrate
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_darkmatterprofile_shape
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_darkmatterprofile_shapegrowthrate
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_disk_angularmomentum
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_disk_halfmassradius
  Code lines: 13
19. Source Code Documentation

**Contained by:** file `objects.nodes.bindings.C.F90`
**Modules used:** `galacticus_nodes` `iso_c_binding`

**function:** `cnode_component_disk_massgas`
**Code lines:** 13
**Contained by:** file `objects.nodes.bindings.C.F90`
**Modules used:** `galacticus_nodes` `iso_c_binding`

**function:** `cnode_component_disk_massstellar`
**Code lines:** 13
**Contained by:** file `objects.nodes.bindings.C.F90`
**Modules used:** `galacticus_nodes` `iso_c_binding`

**function:** `cnode_component_disk_radius`
**Code lines:** 13
**Contained by:** file `objects.nodes.bindings.C.F90`
**Modules used:** `galacticus_nodes` `iso_c_binding`

**function:** `cnode_component_disk_starformationrate`
**Code lines:** 13
**Contained by:** file `objects.nodes.bindings.C.F90`
**Modules used:** `galacticus_nodes` `iso_c_binding`

**function:** `cnode_component_disk_velocity`
**Code lines:** 13
**Contained by:** file `objects.nodes.bindings.C.F90`
**Modules used:** `galacticus_nodes` `iso_c_binding`

**function:** `cnode_component_formationtime_formationtime`
**Code lines:** 13
**Contained by:** file `objects.nodes.bindings.C.F90`
**Modules used:** `galacticus_nodes` `iso_c_binding`

**function:** `cnode_component_hosthistory_hostmassmaximum`
**Code lines:** 13
**Contained by:** file `objects.nodes.bindings.C.F90`
**Modules used:** `galacticus_nodes` `iso_c_binding`

**function:** `cnode_component_hothalo_angularmomentum`
**Code lines:** 13
**Contained by:** file `objects.nodes.bindings.C.F90`
**Modules used:** `galacticus_nodes` `iso_c_binding`

**function:** `cnode_component_hothalo_angularmomentumcold`
**Code lines:** 13
**Contained by:** file `objects.nodes.bindings.C.F90`
**Modules used:** `galacticus_nodes` `iso_c_binding`

**function:** `cnode_component_hothalo_mass`
19.1. Program units

Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes iso_c_binding

function: cnode_component_hothalo_masscold
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_hothalo_outerradius
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_hothalo_outflowedangularmomentum
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_hothalo_outflowedmass
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_hothalo_strippedmass
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_hothalo_trackedoutflowmass
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_hothalo_unaccretedmass
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_interoutput_diskstarformationrate
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding

function: cnode_component_interoutput_spheroidstarformationrate
  Code lines: 13
  Contained by: file objects.nodes.bindings.C.F90
  Modules used: galacticus_nodes iso_c_binding
function: cnode_component_massflowstatistics_cooledmass
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_mergingstatistics_galaxymajormergertime
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_mergingstatistics_nodeformationtime
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_mergingstatistics_nodemajormergertime
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_nbody_velocitydispersion
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_nbody_velocitymaximum
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_satellite_boundmass
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_satellite_mergetime
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_satellite_tidalheatingnormalized
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
   Modules used: galacticus_nodes iso_c_binding

function: cnode_component_satellite_timeofmerging
   Code lines: 13
   Contained by: file objects.nodes.bindings.C.F90
19.1. Program units

Modules used: galacticus_nodesiso_c_binding

function: cnode_component_spheroid_angularmomentum
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodeiso_c_binding

function: cnode_component_spheroid_halfmassradius
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodesiso_c_binding

function: cnode_component_spheroid_massgas
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodesiso_c_binding

function: cnode_component_spheroid_massstellar
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodesiso_c_binding

function: cnode_component_spheroid_radius
Code lines: 13
Containned by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodesiso_c_binding

function: cnode_component_spheroid_starformationrate
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodesiso_c_binding

function: cnode_component_spheroid_velocity
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodesiso_c_binding

function: cnode_component_spin_spin
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodesiso_c_binding

function: cnode_component_spin_spingrowthrate
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodesiso_c_binding

file: objects.nodes.components.F90
Description: Contains a module which implements top-level functions for node components.
19. Source Code Documentation

**module:** node_components

*Description:* Implements top-level functions for node components.

*Code lines:* 48

*Contained by:* file objects.nodes.components.F90

*Used by:* function galacticus_task_evolve_tree, subroutine halo_mass_function_compute

**subroutine:** node_components_initialize

*Description:* Perform initialization tasks for node components.

*Code lines:* 39

*Contained by:* module node_components

*Modules used:* node_component_black_hole_standard, node_component_dark_matter_profile-, node_component_disk_exponential, node_component_disk_very_simple, node_component_hot_halo_standard, node_component_satellite_standard, node_component_spheroid_standard

**file:** objects.nodes.components.age_statistics.standard.F90

*Description:* Contains a module which implements the standard galaxy age statistics component.

*Code lines:* 196

**module:** node_component_age_statistics_standard

*Description:* Implements the standard galaxy age statistics component.

*Code lines:* 176

*Contained by:* file objects.nodes.components.age_statistics.standard.F90

*Modules used:* galacticus_nodes

*Used by:* subroutine tree_node_compute_derivatives, subroutine tree_node_evolve, subroutine satellite_merger_process

**subroutine:** node_component_age_statistics_standard_rate_compute

*Description:* Compute the exponential disk node mass rate of change.

*Code lines:* 32

*Contained by:* module node_component_age_statistics_standard

*Modules used:* galacticus_nodes, galacticus_output_times

**subroutine:** node_component_age_statistics_standard_satellite_merger

*Description:* Remove any age statistics quantities associated with thisNode and add them to the merge target.

*Code lines:* 52

*Contained by:* module node_component_age_statistics_standard

*Modules used:* galacticus_error, satellite_merging_mass_movements-descendants

**subroutine:** node_component_age_statistics_standard_scale_set

*Description:* Set scales for properties of thisNode.

*Code lines:* 27

*Contained by:* module node_component_age_statistics_standard
19.1. Program units

file: objects.nodes.components.basic.non_evolving.F90
Description: Contains a module with the standard implementation of basic tree node methods.
Code lines: 130

module: node_component_basic_non_evolving
Description: A non-evolving implementation of basic tree node methods.
Code lines: 110
Contained by: file objects.nodes.components.basic.non_evolving.F90
Modules used: galacticus_nodes
Used by: subroutine tree_node_compute_derivatives subroutine tree_node_evolve subroutine tree_node_promote

subroutine: node_component_basic_non_evolving_promote
Description: Ensure that thisNode is ready for promotion to its parent. In this case, we simply update the mass of thisNode to be that of its parent.
Code lines: 23
Contained by: module node_component_basic_non_evolving
Modules used: galacticus_error

subroutine: node_component_basic_non_evolving_rate_compute
Description: Compute rates of change of properties in the standard implementation of the basic component.
Code lines: 17
Contained by: module node_component_basic_non_evolving

subroutine: node_component_basic_non_evolving_scale_set
Description: Set scales for properties in the standard implementation of the basic component.
Code lines: 16
Contained by: module node_component_basic_non_evolving

file: objects.nodes.components.basic.standard.F90
Description: Contains a module with the standard implementation of basic tree node methods.
Code lines: 257

module: node_component_basic_standard
Description: The standard implementation of basic tree node methods.
Code lines: 237
Contained by: file objects.nodes.components.basic.standard.F90
Modules used: galacticus_nodes
Used by: subroutine events_node_merger subroutine tree_node_compute_derivatives subroutine tree_node_evolve subroutine tree_node_promote subroutine merger_tree_initialize

subroutine: node_component_basic_standard_promote
Description: Ensure that thisNode is ready for promotion to its parent. In this case, we simply update the mass of thisNode to be that of its parent.
Code lines: 25
19. Source Code Documentation

Contained by: module node_component_basic_standard
Modules used: galacticus_error

subroutine: node_component_basic_standard_rate_compute
Description: Compute rates of change of properties in the standard implementation of the basic component.
Code lines: 19
Contained by: module node_component_basic_standard

subroutine: node_component_basic_standard_scale_set
Description: Set scales for properties in the standard implementation of the basic component.
Code lines: 19
Contained by: module node_component_basic_standard

subroutine: node_component_basic_standard_stop_accretion
Description: Switch off accretion of new mass onto this node once it becomes a satellite.
Code lines: 17
Contained by: module node_component_basic_standard

subroutine: node_component_basic_standard_tree_initialize
Description: Set the mass accretion rate for thisNode.
Code lines: 66
Contained by: module node_component_basic_standard

function: node_component_basic_standard_unresolved_mass
Description: Return the unresolved mass for thisNode.
Code lines: 19
Contained by: module node_component_basic_standard

file: objects.nodes.components.basic.standard_tracking.F90
Description: Contains a module which extends the standard implementation of basic component to track the maximum progenitor mass.
Code lines: 102

module: node_component_basic_standard_tracking
Description: Extends the standard implementation of basic component to track the maximum progenitor mass.
Code lines: 82
Contained by: file objects.nodes.components.basic.standard_tracking.F90
Modules used: galacticus_nodes
Used by: subroutine tree_node_promote subroutine merger_tree_initialize

subroutine: node_component_basic_standard_tracking_promote
Description: Ensure that thisNode is ready for promotion to its parent. In this case, we simply update the maximum mass of thisNode to be that of its parent.
Code lines: 21
Contained by: module node_component_basic_standard_tracking
Modules used: galacticus_error

subroutine: node_component_basic_standard_tree_tracking_initialize
19.1. Program units

*Description:* Set the mass accretion rate for `thisNode`.
*Code lines:* 24
*Contained by:* module `node_component_basic_standard_tracking`

**file:** `objects.nodes.components.black_hole.simple.F90`
*Description:* Contains a module which implements the simple black hole node component.
*Code lines:* 436

**module:** `node_component_black_hole_simple`
*Description:* Implements the simple black hole node component.
*Code lines:* 416
*Contained by:* file `objects.nodes.components.black_hole.simple.F90`
*Modules used:* `galacticus_nodes`
*Used by:* subroutine `count_properties` subroutine `establish_property_names` subroutine `galacticus_merger_tree_output` subroutine `tree_node_compute_derivatives` subroutine `tree_node_evolve` subroutine `satellite_merger_process`

**subroutine:** `node_component_black_hole_simple_create`
*Description:* Creates a simple black hole component for `thisNode`.
*Code lines:* 11
*Contained by:* module `node_component_black_hole_simple`

**subroutine:** `node_component_black_hole_simple_initialize`
*Description:* Initializes the simple black hole node component module.
*Code lines:* 109
*Contained by:* module `node_component_black_hole_simple`
*Modules used:* `input_parameters`

**function:** `node_component_black_hole_simple_matches`
*Description:* Return true if the black hole component of `thisNode` is a match to the simple implementation.
*Code lines:* 17
*Contained by:* module `node_component_black_hole_simple`

**subroutine:** `node_component_black_hole_simple_output`
*Description:* Store black hole properties in the GALACTICUS output file buffers.
*Code lines:* 27
*Contained by:* module `node_component_black_hole_simple`
*Modules used:* `kind_numbers`

**subroutine:** `node_component_black_hole_simple_output_count`
*Description:* Account for the number of black hole properties to be written to the GALACTICUS output file.
*Code lines:* 13
*Contained by:* module `node_component_black_hole_simple`

**subroutine:** `node_component_black_hole_simple_output_names`
*Description:* Set names of black hole properties to be written to the GALACTICUS output file.
19. Source Code Documentation

**Code lines:** 33  
**Contained by:** module `node_component_black_hole_simple`  
**Modules used:** `numerical_constants_astronomical`

**subroutine:** `node_component_black_hole_simple_rate_compute`  
**Description:** Compute the black hole mass rate of change.  
**Code lines:** 76  
**Contained by:** module `node_component_black_hole_simple`  
**Modules used:** `cooling_radii` `dark_matter_halo_scales` `numerical_constants_physical`

**subroutine:** `node_component_black_hole_simple_satellite_merging`  
**Description:** Merge (instantaneously) any simple black hole associated with `thisNode` before it merges with its host halo.  
**Code lines:** 23  
**Contained by:** module `node_component_black_hole_simple`  
**Modules used:** `black_hole_binary_mergers`

**subroutine:** `node_component_black_hole_simple_scale_set`  
**Description:** Set scales for properties of `thisNode`.  
**Code lines:** 18  
**Contained by:** module `node_component_black_hole_simple`

**file:** `objects.nodes.components.black_hole.simple.structure.F90`  
**Description:** Contains a module which implements the structure tasks for the simple black hole node component.  
**Code lines:** 136

**module:** `node_component_black_hole_simple_structure`  
**Description:** Implements the structure tasks for the simple black hole node component.  
**Code lines:** 116  
**Contained by:** file `objects.nodes.components.black_hole.simple.structure.F90`  
**Used by:** function `galactic_structure_potential` subroutine `galactic_structure_radii_initial_adiabatic_compute_factors` function `galactic_structure_rotation_curve` function `galactic_structure_rotation_curve_gradient`  
**Modules used:** `black_hole_fundamentals` `galactic_structure_options` `galacticus_nodes` `numerical_constants_physical`

**function:** `node_component_black_hole_simple_potential`  
**Description:** Compute the gravitational potential due to a black hole.  
**Code lines:** 28  
**Contained by:** module `node_component_black_hole_simple_structure`  
**Modules used:** `black_hole_fundamentals` `galactic_structure_options` `galacticus_nodes` `numerical_constants_physical`

**function:** `node_component_black_hole_simple_rotation_curve`  
**Description:** Computes the rotation curve for the central black hole. Assumes a point mass black hole with a Keplerian rotation curve, except that the rotation speed is limited to never exceed the speed of light.  
**Code lines:** 32  
**Contained by:** module `node_component_black_hole_simple_structure`  
**Modules used:** `black_hole_fundamentals` `galactic_structure_options`
19.1. Program units

**galacticus_nodes**  **numerical_constants_physical**

**function:** node_component_black_hole_simple_rotation_curve_gradient  
**Description:** Computes the rotation curve gradient for the central black hole. Assumes a point mass black hole with a Keplerian rotation curve, *except* that the rotation speed is limited to never exceed the speed of light.  
**Code lines:** 33  
**Contained by:** module node_component_black_hole_simple_structure  
**Modules used:**  
- black_hole_fundamentals  
- galactic_structure_options  
- galacticus_nodes  
- numerical_constants_physical

**file:** objects.nodes.components.black_hole.standard.F90  
**Description:** Contains a module which implements the standard black hole node component.  
**Code lines:** 1221

**module:** node_component_black_hole_standard  
**Description:** Implement black hole tree node methods.  
**Code lines:** 1201  
**Contained by:** file objects.nodes.components.black_hole.standard.F90  
**Modules used:** galacticus_nodes  
**Used by:**  
- subroutine count_properties  
- subroutine establish_property_names  
- subroutine galacticus_merger_tree_output  
- subroutine tree_node_compute_derivatives  
- subroutine tree_node_evolve  
- subroutine satellite_merger_process  
- subroutine node_components_initialize

**function:** hot_mode_fraction  
**Description:** A simple interpolating function which is used as a measure of the fraction of a halo which is in the hot accretion mode.  
**Code lines:** 22  
**Contained by:** module node_component_black_hole_standard  
**Modules used:** cooling_radii  
- dark_matter_halo_scales

**function:** node_component_black_hole_standard_accretion_rate  
**Description:** Return the rest mass accretion rate onto a standard black hole.  
**Code lines:** 9  
**Contained by:** module node_component_black_hole_standard

**subroutine:** node_component_black_hole_standard_create  
**Description:** Creates a black hole component for thisNode.  
**Code lines:** 13  
**Contained by:** module node_component_black_hole_standard

**subroutine:** node_component_black_hole_standard_initialize  
**Description:** Initializes the standard black hole component module.  
**Code lines:** 177  
**Contained by:** module node_component_black_hole_standard  
**Modules used:** input_parameters
subroutine: node_component_black_hole_standard_mass_accretion_rate
Description: Returns the rate of mass accretion onto the black hole in thisNode.
Code lines: 115
Contained by: module node_component_black_hole_standard
Modules used: accretion_disks black_hole_binary_separations
black_hole_fundamentals bondi_hoyle_lyttleton_accretion
galactic_structure_densities galactic_structure_options
hot_halo_temperature_profile ideal_gases_thermodynamics
numerical_constants_astronomical

function: node_component_black_hole_standard_matches
Description: Return true if the black hole component of thisNode is a match to the standard implementation.
Code lines: 17
Contained by: module node_component_black_hole_standard

subroutine: node_component_black_hole_standard_merge_black_holes
Description: Merge two black holes.
Code lines: 42
Contained by: module node_component_black_hole_standard
Modules used: black_hole_binary_mergers black_hole_binary_recoil_velocities
galactic_structure_options galactic_structure_potentials

subroutine: node_component_black_hole_standard_output
Description: Store black hole properties in the GALACTICUS output file buffers.
Code lines: 33
Contained by: module node_component_black_hole_standard
Modules used: accretion_disks kind_numbers

subroutine: node_component_black_hole_standard_output_count
Description: Account for the number of black hole properties to be written to the GALACTICUS output file.
Code lines: 13
Contained by: module node_component_black_hole_standard

subroutine: node_component_black_hole_standard_output_merger
Description: Outputs properties of merging black holes.
Code lines: 30
Contained by: module node_component_black_hole_standard
Modules used: galacticus_hdf5

subroutine: node_component_black_hole_standard_output_names
Description: Set names of black hole properties to be written to the GALACTICUS output file.
Code lines: 72
Contained by: module node_component_black_hole_standard
Modules used: numerical_constants_astronomical

subroutine: node_component_black_hole_standard_output_properties
19.1. Program units

**Description:** Output properties for all black holes in thisNode.
**Code lines:** 89
**Contained by:** module node_component_black_hole_standard
**Modules used:** accretion_disks black_hole_binary_separations
galacticus_hdf5 iso_c_binding
iso_varying_string kind_numbers
memory_management string_handling

**subroutine:** node_component_black_hole_standard_post_evolve
**Description:** Keep black hole spin in physical range.
**Code lines:** 24
**Contained by:** module node_component_black_hole_standard

**function:** node_component_black_hole_standard_radiative_efficiency
**Description:** Return the radiative efficiency of a standard black hole.
**Code lines:** 8
**Contained by:** module node_component_black_hole_standard
**Modules used:** accretion_disks

**subroutine:** node_component_black_hole_standard_rate_compute
**Description:** Compute the black hole node mass rate of change.
**Code lines:** 175
**Contained by:** module node_component_black_hole_standard
**Modules used:** accretion_disks black_hole_binary_separations
cosmology_parameters dark_matter_halo_scales
numerical_constants_astronomical numerical_constants_physical

**subroutine:** node_component_black_hole_standard_satellite_merging
**Description:** Merge any black hole associated with thisNode before it merges with its host halo.
**Code lines:** 75
**Contained by:** module node_component_black_hole_standard
**Modules used:** black_hole_binary_initial_radii black_hole_binary_mergers
black_hole_binary_recoil_velocities galactic_structure_options
galactic_structure_potentials

**subroutine:** node_component_black_hole_standard_scale_set
**Description:** Set scales for properties of thisNode.
**Code lines:** 29
**Contained by:** module node_component_black_hole_standard

**subroutine:** node_component_black_hole_standard_triple_interaction
**Description:** Handles triple black holes interactions, using conditions similar to those of Volonteri et al. [2003].
**Code lines:** 77
**Contained by:** module node_component_black_hole_standard
**Modules used:** galactic_structure_options galactic_structure_potentials
numerical_constants_physical

**file:** objects.nodes.components.black_hole.standard.structure_tasks.F90
19. Source Code Documentation

**Description:**
Contains a module which implements galactic structure tasks for the standard black hole node component.

**Code lines:** 137

**module:** node_component_black_hole_standard_structure_tasks
- **Description:** Implements galactic structure tasks for the standard black hole tree node component.
- **Code lines:** 115
- **Contained by:** file objects.nodes.components.black_hole.standard.structure_tasks.F90
- **Used by:**
  - function galactic_structure_potential
  - subroutine galactic_structure_radii_initial_adiabatic_compute_factors
  - function galactic_structure_rotation_curve
  - function galactic_structure_rotation_curve_gradient

**function:** node_component_black_hole_standard_potential
- **Description:** Compute the gravitational potential due to a black hole.
- **Code lines:** 27
- **Contained by:** module node_component_black_hole_standard_structure_tasks
- **Modules used:**
  - black_hole_fundamentals
galactic_structure_options
galacticus_nodes
numerical_constants_physical

**function:** node_component_black_hole_standard_rotation_curve
- **Description:** Computes the rotation curve for the central black hole. Assumes a point mass black hole with a Keplerian rotation curve, except that the rotation speed is limited to never exceed the speed of light.
- **Code lines:** 32
- **Contained by:** module node_component_black_hole_standard_structure_tasks
- **Modules used:**
  - black_hole_fundamentals
galactic_structure_options
galacticus_nodes
numerical_constants_physical

**function:** node_component_black_hole_standard_rotation_curve_gradient
- **Description:** Computes the rotation curve gradient for the central black hole. Assumes a point mass black hole with a Keplerian rotation curve, except that the rotation speed is limited to never exceed the speed of light.
- **Code lines:** 33
- **Contained by:** module node_component_black_hole_standard_structure_tasks
- **Modules used:**
  - black_hole_fundamentals
galactic_structure_options
galacticus_nodes
numerical_constants_physical

**file:** objects.nodes.components.dark_matter_profile.scale.F90
- **Description:** Contains a module which implements a dark matter profile method that provides a scale radius.
- **Code lines:** 339

**module:** node_component_dark_matter_profile_scale
- **Description:** Implements a dark matter profile method that provides a scale radius.
- **Code lines:** 319
- **Contained by:** file objects.nodes.components.dark_matter_profile.scale.F90
- **Modules used:** galacticus_nodes
- **Used by:**
  - subroutine galactic_structure_radii_solve_adiabatic
  - subroutine galactic_structure_radii_solve_fixed
subroutine: node_component_dark_matter_profile_scale_initialize
Description: Initializes the “scale” implementation of the dark matter halo profile component.
Code lines: 50
Contained by: module node_component_dark_matter_profile_scale
Modules used: input_parameters

subroutine: node_component_dark_matter_profile_scale_initialize_scale
Description: Initialize the scale radius of thisNode.
Code lines: 23
Contained by: module node_component_dark_matter_profile_scale
Modules used: dark_matter_halo_scales dark_matter_profile_scales

subroutine: node_component_dark_matter_profile_scale_plausibility
Description: Determines whether the dark matter profile is physically plausible for radius solving tasks.
Code lines: 15
Contained by: module node_component_dark_matter_profile_scale

subroutine: node_component_dark_matter_profile_scale_promote
Description: Ensure that thisNode is ready for promotion to its parent. In this case, we simply update the growth rate of thisNode to be that of its parent.
Code lines: 24
Contained by: module node_component_dark_matter_profile_scale
Modules used: galacticus_error

subroutine: node_component_dark_matter_profile_scale_rate_compute
Description: Compute the rate of change of the scale radius.
Code lines: 26
Contained by: module node_component_dark_matter_profile_scale
Modules used: dark_matter_halo_scales

function: node_component_dark_matter_profile_scale_scale
Description: Return the scale radius in the dark matter halo profile.
Code lines: 15
Contained by: module node_component_dark_matter_profile_scale
Modules used: dark_matter_halo_scales

subroutine: node_component_dark_matter_profile_scale_scale_set
Description: Set scales for properties of thisNode.
Code lines: 15
Contained by: module node_component_dark_matter_profile_scale

subroutine: node_component_dark_matter_profile_scale_tree_initialize
19. Source Code Documentation

**Description:** Initialize the scale radius of thisNode.
**Code lines:** 34
**Contained by:** module `node_component_dark_matter_profile_scale`

**subroutine: node_component_dark_matter_profile_scale_tree_output**
**Description:** Write the scale radius property to a full merger tree output.
**Code lines:** 35
**Contained by:** module `node_component_dark_matter_profile_scale`
**Modules used:** `io_hdf5` `numerical_constants_astronomical`

**file: objects.nodes.components.dark_matter_profile.scale_preset.F90**
**Description:** Contains a module which implements a dark matter profile method that provides a scale radius.
**Code lines:** 156

**module: node_component_dark_matter_profile_scale_preset**
**Description:** Implements a dark matter profile method that provides a scale radius.
**Code lines:** 136
**Contained by:** file `objects.nodes.components.dark_matter_profile.scale_preset.F90`
**Modules used:** `galacticus_nodes`
**Used by:** subroutine `tree_node_compute_derivatives` subroutine `tree_node_evolve`
**subroutine: node_component_dark_matter_profile_scale_preset_scale_set**
**Description:** Set scales for properties of thisNode.
**Code lines:** 15
**Contained by:** module `node_component_dark_matter_profile_scale_preset`

**subroutine: node_component_dark_matter_profile_scale_preset_rate_compute**
**Description:** Compute the rate of change of the scale radius.
**Code lines:** 17
**Contained by:** module `node_component_dark_matter_profile_scale_preset`
**Modules used:** `dark_matter_halo_scales`

**subroutine: node_component_dark_matter_profile_scale_preset_scale_set**
**Description:** Set scales for properties of thisNode.
**Code lines:** 15
**Contained by:** module `node_component_dark_matter_profile_scale_preset`

**subroutine: node_component_dark_matter_profile_scale_preset_tree_initialize**
**Description:** Initialize the scale radius of thisNode.
**Code lines:** 29
**Contained by:** module `node_component_dark_matter_profile_scale_preset`

**file: objects.nodes.components.dark_matter_profile.scale_shape.F90**
**Description:** Contains a module which implements a dark matter profile method that provides a scale radius and a shape parameter.
module: node_component_dark_matter_profile_scale_shape
Description: Implements a dark matter profile method that provides a scale radius and a shape parameter.
Code lines: 230
Contained by: file objects.nodes.components.dark_matter_profile.scale_shape.F90
Modules used: galacticus_nodes
Used by: subroutine tree_node_compute_derivatives subroutine tree_node_promote subroutine merger_tree_initialize subroutine tree_node_evolve
subroutine: node_component_dark_matter_profile_scale_shape_initialize
Description: Initializes the “scale” implementation of the dark matter halo profile component.
Code lines: 25
Contained by: module node_component_dark_matter_profile_scale_shape
Modules used: input_parameters
subroutine: node_component_dark_matter_profile_scale_shape_initialize_shape
Description: Initialize the shape parameter of thisNode.
Code lines: 14
Contained by: module node_component_dark_matter_profile_scale_shape
Modules used: dark_matter_profiles_shapes
subroutine: node_component_dark_matter_profile_scale_shape_promote
Description: Ensure that thisNode is ready for promotion to its parent. In this case, we simply update the growth rate of thisNode to be that of its parent.
Code lines: 24
Contained by: module node_component_dark_matter_profile_scale_shape
Modules used: galacticus_error
subroutine: node_component_dark_matter_profile_scale_shape_rate_compute
Description: Compute the rate of change of the scale radius.
Code lines: 16
Contained by: module node_component_dark_matter_profile_scale_shape
subroutine: node_component_dark_matter_profile_scale_shape_scale_set
Description: Set scales for properties of thisNode.
Code lines: 15
Contained by: module node_component_dark_matter_profile_scale_shape
subroutine: node_component_dark_matter_profile_scale_shape_tree_initialize
Description: Initialize the scale radius of thisNode.
Code lines: 34
Contained by: module node_component_dark_matter_profile_scale_shape
subroutine: node_component_dark_matter_profile_scale_shape_tree_output
Description: Write the scale radius property to a full merger tree output.
Code lines: 32
Contained by: module node_component_dark_matter_profile_scale_shape
Modules used: io_hdf5

file: objects.nodes.components.disk.exponential.F90
Description: Contains a module which implements the exponential disk node component.
Code lines: 1047

module: node_component_disk_exponential
Description: Implements the exponential disk node component.
Code lines: 1027
Contained by: file objects.nodes.components.disk.exponential.F90
Modules used: galacticus_nodes
Used by:
subroutine galactic_structure_radii_solve_adiabatic
subroutine galactic_structure_radii_solve_linear
subroutine galacticus_calculations_reset
derivatives
subroutine tree_node_compute
subroutine galacticus_merger_tree_output
derivatives
subroutine satellite_merge_process
subroutine node_components_initialize

subroutine: node_component_disk_exponential_calculation_reset
Description: Reset exponential disk structure calculations.
Code lines: 8
Contained by: module node_component_disk_exponential
Modules used: node_component_disk_exponential_data

subroutine: node_component_disk_exponential_create
Description: Create properties in an exponential disk component.
Code lines: 36
Contained by: module node_component_disk_exponential
Modules used: galacticus_output_star_formation_histories
histories
stellar_population_properties

subroutine: node_component_disk_exponential_initialize
Description: Initializes the tree node exponential disk methods module.
Code lines: 105
Contained by: module node_component_disk_exponential
Modules used: abundances_structure input_parameters
memory_management node_component_disk_exponential_data
tables

subroutine: node_component_disk_exponential_post_evolve
Description: Trim histories attached to the disk.
Code lines: 104
Contained by: module node_component_disk_exponential
19.1. Program units

Modules used:
- abundances_structure
- dark_matter_halo_scales
- galacticus_display
- galacticus_error
- histories
- iso_varying_string
- stellar_luminosities_structure
- string_handling

subroutine: node_component_disk_exponential_pre_evolve
  Description: Ensure the disk has been initialized.
  Code lines: 15
  Contained by: module node_component_disk_exponential

function: node_component_disk_exponential_radius_solve
  Description: Return the radius of the exponential disk used in structure solvers.
  Code lines: 9
  Contained by: module node_component_disk_exponential

subroutine: node_component_disk_exponential_radius_solve_set
  Description: Set the radius of the exponential disk used in structure solvers.
  Code lines: 10
  Contained by: module node_component_disk_exponential

subroutine: node_component_disk_exponential_radius_solver
  Description: Interface for the size solver algorithm.
  Code lines: 49
  Contained by: module node_component_disk_exponential
  Modules used:
  - node_component_disk_exponential_data
  - numerical_constants_physical
  - tables

subroutine: node_component_disk_exponential_radius_solver_plausibility
  Description: Determines whether the disk is physically plausible for radius solving tasks. Require that it have non-zero mass and angular momentum.
  Code lines: 39
  Contained by: module node_component_disk_exponential
  Modules used:
  - dark_matter_halo_scales

subroutine: node_component_disk_exponential_rate_compute
  Description: Compute the exponential disk node mass rate of change.
  Code lines: 228
  Contained by: module node_component_disk_exponential
  Modules used:
  - abundances_structure
  - dark_matter_halo_scales
  - galactic_dynamics_bar_instabilities
  - galacticus_error
  - galacticus_output_star_formation_histories
  - histories
  - iso_varying_string
  - numerical_constants_astronomical
  - ram_pressure_stripping_mass_loss_rate_disks
  - star_formation_feedback_disks
  - star_formation_feedback_expulsion_disks
  - stellar_luminosities_structure
  - stellar_population_properties
  - tidal_stripping_mass_loss_rate_disks
**Subroutine:** node_component_disk_exponential_satellite_merging  
**Description:** Transfer any exponential disk associated with thisNode to its host halo.  
**Code lines:** 100  
**Contained by:** module node_component_disk_exponential  
**Modules used:** abundances_structure, histories, galacticus_error, satellite_merging_mass_movements_descriptors, stellar_luminosities_structure

**Subroutine:** node_component_disk_exponential_scale_set  
**Description:** Set scales for properties of thisNode.  
**Code lines:** 59  
**Contained by:** module node_component_disk_exponential  
**Modules used:** abundances_structure, galacticus_output_star_formation_histories, histories, stellar_luminosities_structure, stellar_population_properties

**Subroutine:** node_component_disk_exponential_star_formation_history_output  
**Description:** Store the star formation history in the output file.  
**Code lines:** 23  
**Contained by:** module node_component_disk_exponential  
**Modules used:** galacticus_output_star_formation_histories, histories, iso_c_binding, kind_numbers

**Function:** node_component_disk_exponential_star_formation_rate  
**Description:** Return the star formation rate of the exponential disk.  
**Code lines:** 24  
**Contained by:** module node_component_disk_exponential  
**Modules used:** star_formation_timescales_disks

**Function:** node_component_disk_exponential_velocity  
**Description:** Return the circular velocity of the exponential disk.  
**Code lines:** 9  
**Contained by:** module node_component_disk_exponential

**Subroutine:** node_component_disk_exponential_velocity_set  
**Description:** Set the circular velocity of the exponential disk.  
**Code lines:** 10  
**Contained by:** module node_component_disk_exponential

**File:** objects.nodes.components.disk.exponential.data.F90  
**Description:** Contains a module which implements the exponential disk node component.  
**Code lines:** 219

**Module:** node_component_disk_exponential_data
19.1. Program units

**Code lines:** 199  
**Contained by:** file objects.nodes.components.disk.exponential.data.F90  
**Modules used:** kind_numbers tables  
**Used by:** subroutine galacticus_state_retrieve subroutine galacticus_state_store  
function node_component_disk_-exponential_density function node_component_disk_-exponential_enclosed_mass  
function node_component_disk_-exponential_potential function node_component_disk_-exponential_rotation_curve  
function node_component_disk_-exponential_rotation_curve_gradient function node_component_disk_-exponential_surface_density  
subroutine node_component_disk_-exponential_radius_solver  
function node_component_disk_exponential_enclosed_mass_dimensionless  
**Description:** Returns the fractional mass enclosed within radius in a dimensionless exponential disk.  
**Code lines:** 7  
**Contained by:** module node_component_disk_exponential_data  
**subroutine:** node_component_disk_exponential_reset  
**Description:** Reset calculations for the exponential disk component.  
**Code lines:** 11  
**Contained by:** module node_component_disk_exponential_data  
**function:** node_component_disk_exponential_rotation_curve_bessel_factors  
**Description:** Compute Bessel function factors appearing in the expression for an razor-thin exponential disk rotation curve.  
**Code lines:** 48  
**Contained by:** module node_component_disk_exponential_data  
**Modules used:** bessel_functions numerical_constants_math  
**function:** node_component_disk_exponential_rttn_crv_grdnt_bssl_fctrs  
**Description:** Compute Bessel function factors appearing in the expression for a razor-thin exponential disk rotation curve gradient.  
**Code lines:** 50  
**Contained by:** module node_component_disk_exponential_data  
**Modules used:** bessel_functions numerical_constants_math  
**subroutine:** node_component_disk_exponential_state_retrieve  
**Description:** Retrieve the tabulation state from the file.  
**Code lines:** 12  
**Contained by:** module node_component_disk_exponential_data  
**Modules used:** fgsl  
**subroutine:** node_component_disk_exponential_state_store  
**Description:** Write the tabulation state to file.  
**Code lines:** 9  
**Contained by:** module node_component_disk_exponential_data
19. Source Code Documentation

**Modules used:** fsd

**file:** objects.nodes.components.disk.very_simple.F90

*Description:* Contains a module that implements a very simple disk component.

*Code lines:* 323

**module:** node_component_disk_very_simple

*Description:* Implements a very simple disk component.

*Code lines:* 303

*Contained by:* file objects.nodes.components.disk.very_simple.F90

*Modules used:* galacticus_nodes iso_varying_string

*Used by:* subroutine tree_node_compute_derivatives subroutine satellite_merger_process subroutine node_components_initialize

**subroutine:** node_component_disk_very_simple_initialize

*Description:* Initializes the tree node very simple disk component module.

*Code lines:* 39

*Contained by:* module node_component_disk_very_simple

*Modules used:* input_parameters

**subroutine:** node_component_disk_very_simple_post_evolve

*Description:* Catch rounding errors in the very simple disk gas evolution.

*Code lines:* 52

*Contained by:* module node_component_disk_very_simple

*Modules used:* galacticus_display string_handling

**subroutine:** node_component_disk_very_simple_rate_compute

*Description:* Compute the very simple disk node mass rate of change.

*Code lines:* 49

*Contained by:* module node_component_disk_very_simple

*Modules used:* dark_matter_halo_scales star_formation_feedback_disks stellar_feedback

**subroutine:** node_component_disk_very_simple_satellite_merging

*Description:* Transfer any very simple disk associated with thisNode to its host halo.

*Code lines:* 41

*Contained by:* module node_component_disk_very_simple

*Modules used:* galacticus_error satellite_merging_mass_movements_descriptors

**subroutine:** node_component_disk_very_simple_scale_set

*Description:* Set scales for properties of thisNode.

*Code lines:* 19

*Contained by:* module node_component_disk_very_simple

**function:** node_component_disk_very_simple_sfr

*Description:* Return the star formation rate of the very simple disk.

*Code lines:* 31
19.1. Program units

Contained by: module node_component_disk_very_simple
Modules used: dark_matter_halo_scales           star_formation_timescales_disks

file: objects.nodes.components.dynamics_statistics.bars.F90
Description: Contains a module which implements tracking of dynamics statistics related to bars.
Code lines: 266

module: node_component_dynamics_statistics_bars
Description: Implements tracking of dynamics statistics related to bars.
Code lines: 246
Contained by: file objects.nodes.components.dynamics_statistics.bars.F90
Modules used: galacticus_nodes
Used by: subroutine galacticus_merger_tree_output    function time_step_get

subroutine: node_component_dynamics_statistics_bars_output
Description: Store the dynamical histories of galaxies to GALACTICUS output file.
Code lines: 84
Contained by: module node_component_dynamics_statistics_bars
Modules used: galacticus_hdf5           galacticus_nodes
              io_hdf5               iso_c_binding
              iso_varying_string   numerical_constants_astronomical
              string_handling

subroutine: node_component_dynamics_statistics_bars_record
Description: Record the bar dynamical state of a satellite galaxy.
Code lines: 39
Contained by: module node_component_dynamics_statistics_bars
Modules used: galactic_dynamics_bar_instabilities kepler_orbits
              numerical_constants_math numerical_interpolation
              satellite_orbits

subroutine: node_component_dynamics_statistics_bars_timestep
Description: Determines the timestep to go to the next tabulation point for galactic bar dynamics storage.
Code lines: 67
Contained by: module node_component_dynamics_statistics_bars
Modules used: dark_matter_halo_scales          evolve_to_time_reports
              input_parameters           iso_varying_string

file: objects.nodes.components.formation_times.Cole2000.F90
Description: Contains a module of halo formation time methods.
Code lines: 192

module: node_component_formation_times_cole2000
Description: Implement tracking of halo formation times.
Code lines: 172
Contained by: file objects.nodes.components.formation_times.Cole2000.F90
Modules used: galacticus_nodes
19. Source Code Documentation

Used by: subroutine tree_node_compute_derivatives subroutine tree_node_promote subroutine merger_tree_initialize

subroutine: node_component_formation_time_cole2000_create
Description: Creates a halo formation time component for thisNode. This function is also used to “reform” the halo, since it simply resets the formation time and mass to the current values.
Code lines: 30
Contained by: module node_component_formation_times_cole2000
Modules used: events_halo formation

subroutine: node_component_formation_time_cole2000_node_promotion
Code lines: 18
Contained by: module node_component_formation_times_cole2000

subroutine: node_component_formation_time_cole2000_rate_compute
Description: Check for need to update the formation time of a node in the Cole2000 formation time component.
Code lines: 24
Contained by: module node_component_formation_times_cole2000

subroutine: node_component_formation_time_cole2000_tree_initialize
Description: Initialize the formation node pointer for any childless node.
Code lines: 9
Contained by: module node_component_formation_times_cole2000

subroutine: node_component_formation_times_cole2000_initialize
Description: Initializes the tree node formation time tracking module.
Code lines: 37
Contained by: module node_component_formation_times_cole2000
Modules used: input_parameters

file: objects.nodes.components.host_history.standard.F90
Description: Contains a module which implements a component class that tracks the maximum host mass seen by each halo.
Code lines: 98

module: node_component_host_history_standard
Description: Implements a component class that tracks the maximum host mass seen by each halo.
Code lines: 78
Contained by: file objects.nodes.components.host_history.standard.F90
Modules used: galacticus_nodes
Used by: subroutine events_node_merger subroutine tree_node_evolve subroutine tree_node_promote subroutine merger_tree_initialize

subroutine: node_component_host_history_standard_merger_tree_init
Description: Initialize the standard host history component by creating components in nodes and assigning host mass for satellites.
Code lines: 21
Contained by: module node_component_host_history_standard
subroutine: node_component_host_history_standard_update_history
Description: Record any major merger of thisNode.
Code lines: 16
Contained by: module node_component_host_history_standard

file: objects.nodes.components.hot_halo.cold_mode.F90
Description: Contains a module which implements an extension to the standard hot halo node component which supports a cold mode reservoir.
Code lines: 602

module: node_component_hot_halo_cold_mode
Description: Implements an extension to the standard hot halo node component which supports a cold mode reservoir.
Code lines: 581
Contained by: file objects.nodes.components.hot_halo.cold_mode.F90
Modules used: galacticus_nodes iso_varying_string
radiation_structure
Used by: subroutine event_halo_formation subroutine get_tree
subroutine merger_tree_evolve_to subroutine events_node_merger
subroutine tree_node_compute_derivatives subroutine tree_node_evolve
subroutine tree_node_promote subroutine satellite_merger_process
subroutine merger_tree_initialize

subroutine: node_component_hot_halo_cold_mode_formation
Description: Updates the hot halo gas distribution at a formation event, if requested.
Code lines: 26
Contained by: module node_component_hot_halo_cold_mode
Modules used: abundances_structure dark_matter_halo_scales
galacticus_nodes iso_varying_string
radiation_structure
node_component_hot_halo_standard_data numerical_constants_astronomical

subroutine: node_component_hot_halo_cold_mode_initialize
Description: Initializes the standard hot halo component module.
Code lines: 44
Contained by: module node_component_hot_halo_cold_mode
Modules used: abundances_structure input_parameters

subroutine: node_component_hot_halo_cold_mode_node_merger
Description: Starve thisNode by transferring its hot halo to its parent.
Code lines: 60
Contained by: module node_component_hot_halo_cold_mode
Modules used: abundances_structure cosmology_parameters
dark_matter_halo_scales galactic_structure_enclosed_masses
galactic_structure_options node_component_hot_halo_standard_data

subroutine: node_component_hot_halo_cold_mode_outflow_return
Description: Return outflowed gas to the cold mode reservoir.
Code lines: 69
subroutine: node_component_hot_halo_cold_mode_promote
Description: Ensure that thisNode is ready for promotion to its parent. In this case, we simply update the cold mode mass of thisNode to account for any cold mode gas already in the parent.
Code lines: 28
Contained by: module node_component_hot_halo_cold_mode
Modules used: dark_matter_halo_scales

subroutine: node_component_hot_halo_cold_mode_push_to_cooling_pipes
Description: Push mass through the cooling pipes (along with appropriate amounts of metals and angular momentum) at the given rate.
Code lines: 58
Contained by: module node_component_hot_halo_cold_mode
Modules used: abundances_structure cooling_infall_radii cooling_specific-angular momentoa node_component_hot_halo_standard_data

subroutine: node_component_hot_halo_cold_mode_rate_compute
Description: Compute the hot halo node mass rate of change.
Code lines: 78
Contained by: module node_component_hot_halo_cold_mode
Modules used: abundances_structure accretion_halos cooling_cold_mode_infall_rates dark_matter_halo_scales dark_matter_halo_spins galactic_structure_options hot_halo_ram_pressure_stripping node_component_hot_halo_standard_data numerical_constants_astronomical

subroutine: node_component_hot_halo_cold_mode_satellite_merger
Description: Remove any cold mode gas associated with thisNode before it merges with its host halo.
Code lines: 34
Contained by: module node_component_hot_halo_cold_mode
Modules used: abundances_structure dark_matter_halo_scales node_component_hot_halo_standard_data

subroutine: node_component_hot_halo_cold_mode_scale_set
Description: Set scales for properties of thisNode.
Code lines: 31
Contained by: module node_component_hot_halo_cold_mode
Modules used: abundances_structure dark_matter_halo_scales

subroutine: node_component_hot_halo_cold_mode_thread_initialize
Description: Initializes the tree node hot halo methods module.
Code lines: 12
19.1. Program units

**Description:**

Initialize the contents of the hot halo component for any sub-resolution accretion (i.e. the gas that would have been accreted if the merger tree had infinite resolution).

**Code lines:**

34

**Description:**

Contains a module which implements structure tasks for the cold mode hot halo component.

**Code lines:**

175

**Description:**

Implements structure tasks for the cold mode hot halo component.

**Code lines:**

155

**Description:**

Computes the density at a given position for a dark matter profile.

**Code lines:**

40

**Description:**

Computes the mass within a given radius for the cold mode hot halo component.

**Code lines:**

38

**Description:**

Computes the rotation curve gradient at a given radius for the hot halo density profile.

**Code lines:**

24
19. Source Code Documentation

**Modules used:**
- galactic_structure_options
- galacticus_nodes
- numerical_constants_math
- numerical_constants_physical

**function:** node_component_hot_halo_cold_mode_rotation_curve_task
- **Description:** Computes the rotation curve at a given radius for the hot halo density profile.
- **Code lines:** 20
- **Contained by:** module node_component_hot_halo_cold_mode_structure_tasks
- **Modules used:**
  - galactic_structure_options
  - galacticus_nodes
  - numerical_constants_physical

**file:** objects.nodes.components.hot_halo.outflow_tracking.F90
- **Description:** Contains a module which implements an extension of the standard hot halo node component which tracks the metals arriving from outflows.
- **Code lines:** 125

**module:** node_component_hot_halo_outflow_tracking
- **Description:** Implements an extension of the standard hot halo node component which tracks the metals arriving from outflows.
- **Code lines:** 104
- **Contained by:** file objects.nodes.components.hot_halo.outflow_tracking.F90
- **Modules used:** galacticus_nodes
- **Used by:** subroutine tree_node_compute_
- subroutine tree_node_evolve
derivatives

**subroutine:** node_component_hot_halo_outflow_tracking_rate_compute
- **Description:** Compute the hot halo node mass rate of change.
- **Code lines:** 28
- **Contained by:** module node_component_hot_halo_outflow_tracking
- **Modules used:**
  - abundances_structure
  - dark_matter_halo_scales
  - node_component_hot_halo_standard_data

**subroutine:** node_component_hot_halo_outflow_tracking_scale_set
- **Description:** Set scales for properties of thisNode.
- **Code lines:** 26
- **Contained by:** module node_component_hot_halo_outflow_tracking
- **Modules used:**
  - abundances_structure
  - chemical_abundances_structure
  - dark_matter_halo_scales

**file:** objects.nodes.components.hot_halo.standard.F90
- **Description:** Contains a module which implements the standard hot halo node component.
- **Code lines:** 1535

**module:** node_component_hot_halo_standard
- **Description:** Implements the standard hot halo node component.
- **Code lines:** 1515
- **Contained by:** file objects.nodes.components.hot_halo.standard.F90
- **Modules used:** galacticus_nodes
- **Used by:** subroutine event_halo_formation
- subroutine galacticus_calculations_reset

1700
19.1. Program units

subroutine merger_tree_evolve_to subroutine events_node_merger
derivatives subroutine tree_node_evolve
subroutine tree_node_promote subroutine satellite_merger_process
subroutine merger_tree_initialize subroutine node_components_initialize

**subroutine: node_component_hot_halo_standard_cooling_rate**
*Description:* Get and store the cooling rate for thisNode.
*Code lines:* 25
*Contained by:* module node_component_hot_halo_standard
*Modules used:* cooling_rates

**subroutine: node_component_hot_halo_standard_create**
*Description:* Creates a hot halo component for thisNode.
*Code lines:* 14
*Contained by:* module node_component_hot_halo_standard

**subroutine: node_component_hot_halo_standard_formation**
*Description:* Updates the hot halo gas distribution at a formation event, if requested.
*Code lines:* 59
*Contained by:* module node_component_hot_halo_standard
*Modules used:* abundances_structure chemical_abundances_structure
chemical_reaction_rates_utilities chemical_states
dark_matter_halo_scales node_component_hot_halo_standard_data
numerical_constants_astronomical

**subroutine: node_component_hot_halo_standard_heat_source**
*Description:* An incoming pipe for sources of heating to the hot halo.
*Code lines:* 51
*Contained by:* module node_component_hot_halo_standard
*Modules used:* dark_matter_halo_scales galacticus_error

**subroutine: node_component_hot_halo_standard_hot_gas_all_rate**
*Description:* Adjusts the rates of all components of the hot gas reservoir under the assumption of uniformly
distributed properties (e.g. fully-mixed metals).
*Code lines:* 26
*Contained by:* module node_component_hot_halo_standard

**subroutine: node_component_hot_halo_standard_initialize**
*Description:* Initializes the standard hot halo component module.
*Code lines:* 205
*Contained by:* module node_component_hot_halo_standard
*Modules used:* abundances_structure chemical_abundances_structure
galacticus_error input_parameters
iso_varying_string node_component_hot_halo_standard_data

**subroutine: node_component_hot_halo_standard_initializor**
*Description:* Initializes a standard hot halo component.
*Code lines:* 18
19. Source Code Documentation

- **Contained by:** module `node_component_hot_halo_standard`
  **Modules used:** `dark_matter_halo_scales`

**subroutine:** `node_component_hot_halo_standard_mass_sink`
- **Description:** Account for a sink of gaseous material in the standard hot halo hot gas.
- **Code lines:** 17
  - **Contained by:** module `node_component_hot_halo_standard`
  - **Modules used:** `dark_matter_halo_scales`

**subroutine:** `node_component_hot_halo_standard_node_merger`
- **Description:** Starve `thisNode` by transferring its hot halo to its parent.
- **Code lines:** 74
  - **Contained by:** module `node_component_hot_halo_standard`
  - **Modules used:** `abundances_structure chemical_abundances_structure cosmology_parameters dark_matter_halo_scales galactic_structure_enclosed_masses galactic_structure_options node_component_hot_halo_standard_data galacticus_error`

**function:** `node_component_hot_halo_standard_outer_radius`
- **Description:** Return the outer radius in the standard hot halo.
- **Code lines:** 12
  - **Contained by:** module `node_component_hot_halo_standard`
  - **Modules used:** `dark_matter_halo_scales`

**function:** `node_component_hot_halo_standard_outer_radius_growth_rate`
- **Description:** Compute the growth rate of the outer radius of the hot halo.
- **Code lines:** 21
  - **Contained by:** module `node_component_hot_halo_standard`
  - **Modules used:** `hot_halo_ram_pressure_stripping hot_halo_ram_pressure_stripping_timescales`

**subroutine:** `node_component_hot_halo_standard_outflow_return`
- **Description:** Return outflowed gas to the hot halo.
- **Code lines:** 92
  - **Contained by:** module `node_component_hot_halo_standard`
  - **Modules used:** `abundances_structure chemical_abundances_structure chemical_reaction_rates_utilities chemical_states cosmology_parameters dark_matter_halo_scales galacticus_error hot_halo_mass_distributions node_component_hot_halo_standard_data numerical_constants_atomic numerical_constants_astronomical numerical_constants_prefixes`

**function:** `node_component_hot_halo_standard_outflow_stripped_fraction`
- **Description:** Compute the fraction of material outflowing into the hot halo of `thisNode` which is susceptible to being stripped away.
- **Code lines:** 24
  - **Contained by:** module `node_component_hot_halo_standard`
  - **Modules used:** `dark_matter_halo_scales hot_halo_mass_distributions`
subroutine: node_component_hot_halo_standard_outflowing_abundances_rate
Description: Accept outflowing gas abundances from a galaxy and deposit it into the outflowed reservoir.
Code lines: 24
Contained by: module node_component_hot_halo_standard
Modules used: abundances_structure

subroutine: node_component_hot_halo_standard_outflowing_ang_mom_rate
Description: Accept outflowing gas angular momentum from a galaxy and deposit it into the outflowed reservoir.
Code lines: 23
Contained by: module node_component_hot_halo_standard
Modules used: node_component_hot_halo_standard_data

subroutine: node_component_hot_halo_standard_outflowing_mass_rate
Description: Accept outflowing gas from a galaxy and deposit it into the outflowed and stripped reservoirs.
Code lines: 24
Contained by: module node_component_hot_halo_standard

subroutine: node_component_hot_halo_standard_post_evolve
Description: Do processing of the node required after evolution.
Code lines: 63
Contained by: module node_component_hot_halo_standard
Modules used: abundances_structure dark_matter_halo_scales
node_component_hot_halo_standard_data

subroutine: node_component_hot_halo_standard_pre_evolve
Description: Ensure the standard hot halo has been initialized.
Code lines: 15
Contained by: module node_component_hot_halo_standard

subroutine: node_component_hot_halo_standard_promote
Description: Ensure that thisNode is ready for promotion to its parent. In this case, we simply update the hot halo mass of thisNode to account for any hot halo already in the parent.
Code lines: 36
Contained by: module node_component_hot_halo_standard
Modules used: dark_matter_halo_scales

subroutine: node_component_hot_halo_standard_push_from_halo
Description: Push mass from the hot halo into an infinite sink (along with appropriate amounts of metals, chemicals and angular momentum) at the given rate.
Code lines: 37
Contained by: module node_component_hot_halo_standard
Modules used: abundances_structure chemical_abundances_structure dark_matter_halo_scales

subroutine: node_component_hot_halo_standard_push_to_cooling_pipes
Description: Push mass through the cooling pipes (along with appropriate amounts of metals and angular momentum) at the given rate.
Code lines: 76
19. Source Code Documentation

**Contained by:** module `node_component_hot_halo_standard`
**Modules used:**
- `abundances_structure`
- `cooling_infall_radii`
- `cooling_specific-angular_momenta`
- `node_component_hot_halo_standard_data`

**subroutine:** `node_component_hot_halo_standard_rate_compute`
**Description:** Compute the hot halo node mass rate of change.
**Code lines:** 116
**Contained by:** module `node_component_hot_halo_standard`
**Modules used:**
- `abundances_structure`
- `accretion_halos`
- `chemical_abundances_structure`
- `chemical_reaction_rates`
- `chemical_reaction_rates_utilities`
- `chemical_states`
- `cosmology_parameters`
- `dark_matter_halo_scales`
- `dark_matter_halo_spins`
- `hot_halo_mass_distributions`
- `hot_halo_ram_pressure_stripping-timescales`
- `node_component_hot_halo_standard_data`
- `timescales`
- `numerical_constants_astronomical`

**subroutine:** `node_component_hot_halo_standard_reset`
**Description:** Remove memory of stored computed values as we’re about to begin computing derivatives anew.
**Code lines:** 8
**Contained by:** module `node_component_hot_halo_standard`

**subroutine:** `node_component_hot_halo_standard_satellite_merger`
**Description:** Remove any hot halo associated with `thisNode` before it merges with its host halo.
**Code lines:** 46
**Contained by:** module `node_component_hot_halo_standard`
**Modules used:**
- `abundances_structure`
- `chemical_abundances_structure`
- `dark_matter_halo_scales`
- `node_component_hot_halo_standard_data`

**subroutine:** `node_component_hot_halo_standard_scale_set`
**Description:** Set scales for properties of `thisNode`.
**Code lines:** 41
**Contained by:** module `node_component_hot_halo_standard`
**Modules used:**
- `abundances_structure`
- `chemical_abundances_structure`
- `dark_matter_halo_scales`

**subroutine:** `node_component_hot_halo_standard_strip_gas_rate`
**Description:** Add gas stripped from the hot halo to the stripped gas reservoirs under the assumption of uniformly distributed properties (e.g. fully-mixed metals).
**Code lines:** 29
**Contained by:** module `node_component_hot_halo_standard`

**subroutine:** `node_component_hot_halo_standard_thread_initialize`
**Description:** Initializes the tree node hot halo methods module.
**Code lines:** 7
**Contained by:** module `node_component_hot_halo_standard`

**subroutine:** `node_component_hot_halo_standard_tree_initialize`
19.1. Program units

**Description:** Initialize the contents of the hot halo component for any sub-resolution accretion (i.e. the gas that would have been accreted if the merger tree had infinite resolution).

**Code lines:** 44

**Contained by:** module `node_component_hot_halo_standard`

**Modules used:** `abundances_structure`, `accretion_halos`, `chemical_abundances_structure`, `dark_matter_halo_spins`

**file:** `objects.nodes.components.hot_halo.standard.data.F90`

**Description:** Contains a module which provides data for the standard hot halo node component.

**Code lines:** 39

**module:** `node_component_hot_halo_standard_data`

**Description:** Provides data for the standard hot halo node component.

**Code lines:** 19

**Contained by:** file `objects.nodes.components.hot_halo.standard.data.F90`

**Used by:** subroutine `node_component_hot_halo_-cold_mode_formation`
subroutine `node_component_hot_halo_-cold_mode_node_merger`
subroutine `node_component_hot_halo_-cold_mode_outflow_return`
subroutine `node_component_hot_halo_-cold_mode_push_to_cooling_pipes`
subroutine `node_component_hot_halo_-cold_mode_rate_compute`
subroutine `node_component_hot_halo_-cold_mode_satellite_merger`
subroutine `node_component_hot_halo_-cold_mode_outflow_tracking_rate_compute`
subroutine `node_component_hot_halo_-cold_mode_standard_initialize`
subroutine `node_component_hot_halo_-cold_mode_standard_outflow_return`
subroutine `node_component_hot_halo_-cold_mode_standard_outflowing_ang_mom_rate`
subroutine `node_component_hot_halo_-cold_mode_standard_post_evolve`
subroutine `node_component_hot_halo_-cold_mode_standard_push_to_cooling_pipes`
subroutine `node_component_hot_halo_-cold_mode_standard_rate_compute`
subroutine `node_component_hot_halo_-cold_mode_standard_satellite_merger`
subroutine `node_component_hot_halo_-cold_mode_standard_formation`
subroutine `node_component_hot_halo_-cold_mode_standard_node_merger`

**file:** `objects.nodes.components.hot_halo.very_simple.F90`

**Description:** Contains a module which implements a very simple hot halo node component.

**Code lines:** 403

**module:** `node_component_hot_halo_very_simple`

**Description:** Implements a very simple hot halo node component.

**Code lines:** 383

**Contained by:** file `objects.nodes.components.hot_halo.very_simple.F90`

**Modules used:** `galacticus_nodes`

**Used by:** subroutine `galacticus_calculations_-reset`
subroutine `events_node_merger`
subroutine `tree_node_compute_-derivatives`
subroutine `tree_node_evolve`
subroutine `tree_node_promote`
subroutine `satellite_merger_process`
subroutine `merger_tree_initialize`
subroutine: node_component_hot_halo_very_simple_cooling_rate
Description: Get and store the cooling rate for thisNode.
Code lines: 20
Contained by: module node_component_hot_halo_very_simple
Modules used: cooling_rates

subroutine: node_component_hot_halo_very_simple_create
Description: Creates a very simple hot halo component for thisNode.
Code lines: 12
Contained by: module node_component_hot_halo_very_simple

subroutine: node_component_hot_halo_very_simple_initialize
Description: Initializes the very simple hot halo component module.
Code lines: 21
Contained by: module node_component_hot_halo_very_simple

subroutine: node_component_hot_halo_very_simple_node_merger
Description: Starve thisNode by transferring its hot halo to its parent.
Code lines: 20
Contained by: module node_component_hot_halo_very_simple

function: node_component_hot_halo_very_simple_outer_radius
Description: Return the outer radius of the hot halo. Assumes a simple model in which this always equals the virial radius.
Code lines: 10
Contained by: module node_component_hot_halo_very_simple
Modules used: dark_matter_halo_scales

subroutine: node_component_hot_halo_very_simple_outflowing_mass_rate
Description: Accept outflowing gas from a galaxy and deposit it into very simple hot halo.
Code lines: 11
Contained by: module node_component_hot_halo_very_simple

subroutine: node_component_hot_halo_very_simple_post_evolve
Description: Do processing of the node required after evolution.
Code lines: 24
Contained by: module node_component_hot_halo_very_simple

subroutine: node_component_hot_halo_very_simple_promote
Description: Ensure that thisNode is ready for promotion to its parent. In this case, we simply update the hot halo mass of thisNode to account for any hot halo already in the parent.
Code lines: 24
Contained by: module node_component_hot_halo_very_simple

subroutine: node_component_hot_halo_very_simple_push_to_cooling_pipes
Description: Push mass through the cooling pipes at the given rate.
Code lines: 25
Contained by: module node_component_hot_halo_very_simple
19.1. Program units

subroutine: node_component_hot_halo_very_simple_rate_compute
Description: Compute the very simple hot halo component mass rate of change.
Code lines: 18
Contained by: module node_component_hot_halo_very_simple

subroutine: node_component_hot_halo_very_simple_reset
Description: Remove memory of stored computed values as we’re about to begin computing derivatives anew.
Code lines: 7
Contained by: module node_component_hot_halo_very_simple

subroutine: node_component_hot_halo_very_simple_satellite_merger
Description: Remove any hot halo associated with thisNode before it merges with its host halo.
Code lines: 22
Contained by: module node_component_hot_halo_very_simple

subroutine: node_component_hot_halo_very_simple_scale_set
Description: Set scales for properties of thisNode.
Code lines: 22
Contained by: module node_component_hot_halo_very_simple

subroutine: node_component_hot_halo_very_simple_tree_initialize
Description: Initialize the contents of the very simple hot halo component.
Code lines: 43
Contained by: module node_component_hot_halo_very_simple
Modules used: cosmology_parameters

file: objects.nodes.components.indices.standard.F90
Description: Contains a module which implements the standard indices component.
Code lines: 68

module: node_component_indices_standard
Description: Implements the standard indices component.
Code lines: 48
Contained by: file objects.nodes.components.indices.standard.F90
Modules used: galacticus_nodes
Used by: subroutine merger_tree_initialize

subroutine: node_component_indices_standard_merger_tree_init
Description: Initialize the indices component by creating components in nodes and storing indices.
Code lines: 19
Contained by: module node_component_indices_standard

file: objects.nodes.components.interoutput.standard.F90
Description: Contains a module which implements the standard indices component.
Code lines: 197

module: node_component_inter_output_standard
19. Source Code Documentation

**Description:** Implements the standard indices component.

**Code lines:** 177

**Contained by:** file `objects.nodes.components.interoutput.standard.F90`

**Modules used:** galacticus_nodes

**Used by:** subroutine `galacticus_merger_tree_output` subroutine `tree_node_compute_derivatives`

subroutine `tree_node_evolve` subroutine `satellite_merger_process`

**subroutine:** node_component_inter_output_standard_rate_compute

**Description:** Compute the exponential disk node mass rate of change.

**Code lines:** 35

**Contained by:** module `node_component_inter_output_standard`

**Modules used:** galacticus_output_times

**subroutine:** node_component_inter_output_standard_reset

**Description:** Reset interoutput accumulated quantities.

**Code lines:** 19

**Contained by:** module `node_component_inter_output_standard`

**Modules used:** iso_c_binding kind_numbers

**subroutine:** node_component_inter_output_standard_satellite_merger

**Description:** Remove any inter-output quantities associated with thisNode and add them to the merge target.

**Code lines:** 44

**Contained by:** module `node_component_inter_output_standard`

**Modules used:** galacticus_error satellite_merging_mass_movementsDescriptors

**subroutine:** node_component_inter_output_standard_scale_set

**Description:** Set scales for properties of thisNode.

**Code lines:** 25

**Contained by:** module `node_component_inter_output_standard`

**File:** `objects.nodes.components.mass_flow_statistics.standard.F90`

**Description:** Contains a module which implements the standard mass flow statistics component.

**Code lines:** 175

**Module:** node_component_mass_flow_statistics_standard

**Description:** Implements the standard mass flow statistics component.

**Code lines:** 155

**Contained by:** file `objects.nodes.components.mass_flow_statistics.standard.F90`

**Modules used:** galacticus_nodes

**Used by:** subroutine `galacticus_merger_tree_output` subroutine `tree_node_compute_derivatives`

subroutine `tree_node_evolve` subroutine `merger_tree_initialize`

**subroutine:** node_component_mass_flow_statistics_standard_extra_output

**Description:** Reset mass flow statistics at output time.

**Code lines:** 23

1708
19.1. Program units

contained by: module node_component_mass_flow_statistics_standard
modules used: galacticus_nodes iso_c_binding kind_numbers

subroutine: node_component_mass_flow_statistics_standard_initialize
description: Initializes the standard mass flow statistics component.
code lines: 24
contained by: module node_component_mass_flow_statistics_standard
modules used: galacticus_nodes iso_c_binding kind_numbers

subroutine: node_component_mass_flow_statistics_standard_merger_tree_init
description: Initialize the mass flow statistics component by creating components in nodes and computing formation times.
code lines: 20
contained by: module node_component_mass_flow_statistics_standard
modules used: dark_matter_halo_formation_times

subroutine: node_component_mass_flow_statistics_standard_rate_compute
description: Compute rates of change of properties in the standard implementation of the basic component.
code lines: 18
contained by: module node_component_mass_flow_statistics_standard
modules used: cooling_rates

subroutine: node_component_mass_flow_statistics_standard_scale_set
description: Set scales for properties in the standard implementation of the massFlowStatistics component.
code lines: 18
contained by: module node_component_mass_flow_statistics_standard
modules used: galacticus_nodes iso_c_binding

file: objects.nodes.components.merging_statistics.recent.F90
description: Contains a module which implements the recent merging statistics component.
code lines: 344

module: node_component_merging_statistics_recent
description: Implements the recent merging statistics component.
code lines: 324
contained by: file objects.nodes.components.merging_statistics.recent.F90
modules used: galacticus_nodes iso_c_binding
used by: subroutine count_properties subroutine establish_property_names
subroutine galacticus_merger_tree_- subroutine events_node_merger
output subroutine tree_node_promote subroutine merger_tree_initialize

subroutine: node_component_merging_statistics_recent_initialize
description: Initializes the recent merging statistics component.
code lines: 74
contained by: module node_component_merging_statistics_recent
modules used: galacticus_error galacticus_output_times
19. Source Code Documentation

input_parameters iso_varying_string
memory_management

function: node_component_merging_statistics_recent_matches
Description: Return true if the black hole component of thisNode is a match to the standard implementation.
Code lines: 17
Contained by: module node_component_merging_statistics_recent

subroutine: node_component_merging_statistics_recent_merger_tree_init
Description: Initialize the merging statistics component by creating components in nodes.
Code lines: 19
Contained by: module node_component_merging_statistics_recent

subroutine: node_component_merging_statistics_recent_node_merger
Description: Record any major merger of thisNode.
Code lines: 62
Contained by: module node_component_merging_statistics_recent
Modules used: dark_matter_halo_scales galacticus_error
galacticus_output_times iso_c_binding

subroutine: node_component_merging_statistics_recent_node_promotion
Description: Ensure that thisNode is ready for promotion to its parent. In this case, we simply update the node merger time.
Code lines: 15
Contained by: module node_component_merging_statistics_recent

subroutine: node_component_merging_statistics_recent_output
Description: Store black hole properties in the GALACTICUS output file buffers.
Code lines: 22
Contained by: module node_component_merging_statistics_recent
Modules used: galacticus_output_times kind_numbers

subroutine: node_component_merging_statistics_recent_output_count
Description: Account for the number of black hole properties to be written to the GALACTICUS output file.
Code lines: 9
Contained by: module node_component_merging_statistics_recent

subroutine: node_component_merging_statistics_recent_output_names
Description: Set names of black hole properties to be written to the GALACTICUS output file.
Code lines: 30
Contained by: module node_component_merging_statistics_recent

file: objects.nodes.components.merging_statistics.standard.F90
Description: Contains a module which implements the standard merging statistics component.
Code lines: 250

module: node_component_merging_statistics_standard
19.1. Program units

**Description:** Implements the standard merging statistics component.

**Code lines:** 230

**Contained by:** file `objects.nodes.components.merging_statistics.standard.F90`

**Modules used:** `galacticus_nodes`

**Used by:** subroutine `events_node_merger` subroutine `tree_node_promote`
subroutine `satellite_merger_process` subroutine `merger_tree_initialize`

**subroutine:** `node_component_merging_statistics_standard_initialize`

**Description:** Initializes the standard merging statistics component.

**Code lines:** 35

**Contained by:** module `node_component_merging_statistics_standard`

**Modules used:** `input_parameters`

**subroutine:** `node_component_merging_statistics_standard_merger_tree_init`

**Description:** Initialize the merging statistics component by creating components in nodes and computing formation times.

**Code lines:** 49

**Contained by:** module `node_component_merging_statistics_standard`

**subroutine:** `node_component_merging_statistics_standard_merger_tree_init_set`

**Description:** Set the initial properties of the standard merging statistics component in a node.

**Code lines:** 14

**Contained by:** module `node_component_merging_statistics_standard`

**Modules used:** `dark_matter_halo_formation_times`

**subroutine:** `node_component_merging_statistics_standard_node_merger`

**Description:** Record any major merger of `thisNode`.

**Code lines:** 16

**Contained by:** module `node_component_merging_statistics_standard`

**subroutine:** `node_component_merging_statistics_standard_node_promotion`

**Description:** Ensure that `thisNode` is ready for promotion to its parent. In this case, we simply update the node merger time.

**Code lines:** 15

**Contained by:** module `node_component_merging_statistics_standard`

**subroutine:** `node_component_merging_statistics_standard_satellite_merger`

**Description:** Record properties of a merging event for `thisNode`.

**Code lines:** 22

**Contained by:** module `node_component_merging_statistics_standard`

**Modules used:** `satellite_merging_mass_movements_descriptors`

**file:** `objects.nodes.components.nBody.standard.F90`

**Description:** Contains a module with the standard implementation of N-body component method.

**Code lines:** 85

**module:** `node_component_nbody_standard`

**Description:** The standard implementation of N-body component method.
19. Source Code Documentation

- **Code lines:** 65
- **Contained by:** file `objects.nodes.components.nBody.standard.F90`
- **Modules used:** `galacticus_nodes`
- **Used by:** subroutine `tree_node_promote`

**subroutine: node_component_nbody_standard_promote**

- **Description:** Ensure that `thisNode` is ready for promotion to its parent. In this case, we simply update the properties of `thisNode` to be those of its parent.
- **Code lines:** 22
- **Contained by:** module `node_component_nbody_standard`

**file: objects.nodes.components.position.preset.F90**

- **Description:** Contains a module which implements a preset position component.
- **Code lines:** 86

**module: node_component_position_preset**

- **Description:** Implements a preset position component.
- **Code lines:** 66
- **Contained by:** file `objects.nodes.components.position.preset.F90`
- **Modules used:** `galacticus_nodes`
- **Used by:** subroutine `tree_node_promote`

**subroutine: node_component_position_preset_node_promotion**

- **Description:** Ensure that `thisNode` is ready for promotion to its parent. In this case, update the position of `thisNode` to that of the parent.
- **Code lines:** 19
- **Contained by:** module `node_component_position_preset`

**file: objects.nodes.components.satellite.orbiting.F90**

- **Description:** Contains a module of satellite orbit tree node methods.
- **Code lines:** 351

**module: node_component_satellite_orbiting**

- **Description:** Implements the orbiting satellite component.
- **Code lines:** 330
- **Contained by:** file `objects.nodes.components.satellite.orbiting.F90`
- **Modules used:** `galacticus_nodes` `kepler_orbits` `tensors`
- **Used by:** subroutine `get_tree` subroutine `events_node_merger` subroutine `tree_node_compute_derivatives` subroutine `tree_node_evolve` subroutine `merger_tree_initialize`

**subroutine: node_component_satellite_orbiting_create**

- **Description:** Create a satellite orbit component and assign initial position, velocity, orbit, and tidal heating quantities. (The initial bound mass is automatically set to the original halo mass by virtue of that being the class default).
- **Code lines:** 61
- **Contained by:** module `node_component_satellite_orbiting`
- **Modules used:** `numerical_constants_math` `pseudo_random` `satellite_merging_timescales` `vectors`
virial_orbits

**subroutine:** node_component_satellite_orbiting_initialize

*Description:* Initializes the orbiting satellite methods module.

*Code lines:* 25

*Contained by:* module node_component_satellite_orbiting

*Modules used:* input_parameters

**subroutine:** node_component_satellite_orbiting_rate_compute

*Description:* Compute rate of change for satellite properties.

*Code lines:* 81

*Contained by:* module node_component_satellite_orbiting

*Modules used:*
- dark_matter_halo_scales
- galactic_structure_enclosed_masses
- galactic_structure_options
- numerical_constants_astronomical
- numerical_constants_physical
- numerical_constants_prefixes
- satellite_dynamical_friction
- satellite_tidal_heating
- satellite_tidal_stripping
- vectors

**subroutine:** node_component_satellite_orbiting_scale_set

*Description:* Set scales for properties of thisNode.

*Code lines:* 32

*Contained by:* module node_component_satellite_orbiting

*Modules used:*
- dark_matter_halo_scales
- galactic_structure_enclosed_masses
- galactic_structure_options
- numerical_constants_astronomical
- numerical_constants_physical
- numerical_constants_prefixes

**subroutine:** node_component_satellite_orbiting_tree_initialize

*Description:* Initialize the orbiting satellite component.

*Code lines:* 7

*Contained by:* module node_component_satellite_orbiting

**subroutine:** node_component_satellite_orbiting_trigger_merger

*Description:* Trigger a merger of the satellite by setting the time until merging to zero.

*Code lines:* 9

*Contained by:* module node_component_satellite_orbiting

**file:** objects.nodes.components.satellite.preset.F90

*Description:* Contains a module which implements a preset satellite orbit component.

*Code lines:* 111

**module:** node_component_satellite_preset

*Description:* Implements a preset satellite orbit component.

*Code lines:* 89

*Contained by:* file objects.nodes.components.satellite.preset.F90

*Modules used:* galacticus_nodes

*Used by:* subroutine tree_node_promote

**subroutine:** node_component_satellite_preset_promote

*Description:* Ensure that thisNode is ready for promotion to its parent. In this case, we simply copy any preset satellite orbit from the parent.
19. Source Code Documentation

Code lines: 13  
Contained by: module node_component_satellite_preset

file: objects.nodes.components.satellite.standard.F90  
Description: Contains a module of satellite orbit tree node methods.  
Code lines: 332

module: node_component_satellite_standard  
Description: Implements the standard satellite component.  
Code lines: 313  
Contained by: file objects.nodes.components.satellite.standard.F90  
Modules used: galacticus_nodes kepler_orbits  
Used by: subroutine event_halo_formation subroutine events_node_merger  
subroutine tree_node_compute_derivatives subroutine tree_node_evolve  
subroutine merger_tree_initialize subroutine node_components_initialize  
subroutine satellite_move_to_new_host

subroutine: node_component_satellite_standard_create  
Description: Create a satellite orbit component and assign a time until merging and a bound mass equal  
initially to the total halo mass.  
Code lines: 52  
Contained by: module node_component_satellite_standard  
Modules used: satellite_merging_timescales virial_orbits

subroutine: node_component_satellite_standard_halo_formation_task  
Description: Reset the orbits of satellite galaxies on halo formation events.  
Code lines: 21  
Contained by: module node_component_satellite_standard

subroutine: node_component_satellite_standard_initialize  
Description: Initializes the standard satellite orbit component module.  
Code lines: 43  
Contained by: module node_component_satellite_standard  
Modules used: input_parameters

subroutine: node_component_satellite_standard_rate_compute  
Description: Compute the time until satellite merging rate of change.  
Code lines: 22  
Contained by: module node_component_satellite_standard  
Modules used: dark_matter_halos_mass_loss_rates

subroutine: node_component_satellite_standard_scale_set  
Description: Set scales for properties of thisNode.  
Code lines: 21  
Contained by: module node_component_satellite_standard

subroutine: node_component_satellite_standard_tree_initialize  
Description: Initialize the standard satellite component.
19.1. Program units

```plaintext
Code lines: 7
Contained by: module node_component_satellite_standard

function: node_component_satellite_standard_virial_orbit
Description: Return the orbit of the satellite at the virial radius.
Code lines: 20
Contained by: module node_component_satellite_standard
Modules used: virial_orbits

subroutine: node_component_satellite_standard_virial_orbit_set
Description: Set the orbit of the satellite at the virial radius.
Code lines: 26
Contained by: module node_component_satellite_standard
Modules used: satellite_merging_timescales

file: objects.nodes.components.satellite.very_simple.F90
Description: Contains a module which implements a very simple satellite orbit component.
Code lines: 179

module: node_component_satellite_very_simple
Description: Implements a very simple satellite orbit component.
Code lines: 159
Contained by: file objects.nodes.components.satellite.very_simple.F90
Modules used: galacticus_nodes Used by: subroutine event_halo_formation subroutine events_node_merger subroutine merger_tree_initialize subroutine satellite_move_to_new_host

subroutine: node_component_satellite_very_simple_create
Description: Create a satellite orbit component and assign a time until merging and a bound mass equal initially to the total halo mass.
Code lines: 42
Contained by: module node_component_satellite_very_simple
Modules used: kepler_orbits satellite_merging_timescales virial_orbits

subroutine: node_component_satellite_very_simple_halo_formation_task
Description: Reset the orbits of satellite galaxies on halo formation events.
Code lines: 20
Contained by: module node_component_satellite_very_simple

subroutine: node_component_satellite_very_simple_initialize
Description: Initializes the tree node satellite orbit methods module.
Code lines: 25
Contained by: module node_component_satellite_very_simple
Modules used: input_parameters

subroutine: node_component_satellite_very_simple_tree_initialize
Description: Initialize the very simple satellite component.
Code lines: 7
```
19. Source Code Documentation

**Contained by:** module `node_component_satellite_very_simple`

**file:** `objects.nodes.components.spheroid.standard.F90`

**Description:** Contains a module which implements the standard spheroid component.

**Code lines:** 1171

**module:** `node_component_spheroid_standard`

**Description:** Implements the standard spheroid component.

**Code lines:** 1151

**Contained by:** file `objects.nodes.components.spheroid.standard.F90`

**Modules used:**
- `galacticus_nodes`
- `node_component_spheroid_standard_data`
- `stellar_population_properties`
- `histories`
- `galactic_structure_radii_adiabatic`
- `galactic_structure_radii.solve_fixed`
- `galactic_structure_radii.solve_linear`
- `galactic_structure_radii.solve_simple`
- `galacticus_merger_tree.output`
- `tree_node_compute.derivatives`
- `tree_node_evolve`
- `satellite_merger_process`
- `node_components_initialize`

**subroutine:** `node_component_spheroid_standard_energy_gas_input_rate`

**Description:** Handles input of energy into the spheroid gas from other components (e.g. black holes). The energy input rate should be in units of $M_\odot km^2 s^{-2} Gyr^{-1}$.

**Code lines:** 44

**Contained by:** module `node_component_spheroid_standard`

**Modules used:**
- `abundances_structure`
- `galacticus_error`

**subroutine:** `node_component_spheroid_standard_initialize`

**Description:** Initializes the tree node standard spheroid methods module.

**Code lines:** 125

**Contained by:** module `node_component_spheroid_standard`

**Modules used:**
- `abundances_structure`
- `galacticus_error`
- `input_parameters`
- `iso_varying_string`
- `memory_management`
- `stellar_luminosities_structure`

**subroutine:** `node_component_spheroid_standard_initializor`

**Description:** Initializes a standard spheroid component.

**Code lines:** 33

**Contained by:** module `node_component_spheroid_standard`

**Modules used:**
- `galacticus_output_star_formation.histories`

**subroutine:** `node_component_spheroid_standard_mass_gas_sink_rate`

**Description:** Account for a sink of gaseous material in the standard spheroid.

**Code lines:** 26

**Contained by:** module `node_component_spheroid_standard`

**Modules used:**
- `abundances_structure`
- `galacticus_error`
19.1. Program units

subroutine: node_component_spheroid_standard_post_evolve
Description: Trim histories attached to the spheroid.
Code lines: 77
Contained by: module node_component_spheroid_standard
Modules used:
- abundances_structure
- galacticus_display
- iso_varying_string
- stellar_luminosities_structure
- string_handling

subroutine: node_component_spheroid_standard_pre_evolve
Description: Ensure the spheroid has been initialized.
Code lines: 15
Contained by: module node_component_spheroid_standard

function: node_component_spheroid_standard_radius_solve
Description: Return the circular radius of the standard spheroid.
Code lines: 9
Contained by: module node_component_spheroid_standard

subroutine: node_component_spheroid_standard_radius_solve_set
Description: Set the scale radius of the standard spheroid.
Code lines: 10
Contained by: module node_component_spheroid_standard

subroutine: node_component_spheroid_standard_radius_solver
Description: Interface for the size solver algorithm.
Code lines: 41
Contained by: module node_component_spheroid_standard

subroutine: node_component_spheroid_standard_radius_solver_plausibility
Description: Determines whether the spheroid is physically plausible for radius solving tasks. Require that it have non-zero mass and angular momentum.
Code lines: 22
Contained by: module node_component_spheroid_standard

subroutine: node_component_spheroid_standard_rate_compute
Description: Compute the standard spheroid node mass rate of change.
Code lines: 156
Contained by: module node_component_spheroid_standard
Modules used:
- abundances_structure
- dark_matter_halo_scales
- galactic_structure_options
- galacticus_output_star_formation_histories
- numerical_constants_astronomical
- ram_pressure_stripping_mass_loss_rate_spheroids
- satellites_tidal_fields
- star_formation_feedback_expulsion_spheroids
- star_formation_feedback_spheroids
- star_formation_feedback_spheroids
- tidal_stripping_mass_loss_rate_spheroids
- stellar_luminosities_structure
subroutine: node_component_spheroid_standard_satellite_merging

Description: Transfer any standard spheroid associated with thisNode to its host halo.

Code lines: 210

Modules used: node_component_spheroid_standard abundances_structure galacticus_error

satellite_merging_mass_movements_-

descriptors

stellar_luminosities_structure

subroutine: node_component_spheroid_standard_scale_set

Description: Set scales for properties of thisNode. Note that gas masses get an additional scaling down since they can approach zero and we’d like to prevent them from becoming negative.

Code lines: 61

Modules used: node_component_spheroid_standard abundances_structure galacticus_output_star_formation_-

histories

stellar_luminosities_structure

subroutine: node_component_spheroid_standard_star_formation_history_extend

Description: Extend the range of a star formation history in a standard spheroid component for thisNode.

Code lines: 15

Modules used: node_component_spheroid_standard

subroutine: node_component_spheroid_standard_star_formation_history_output

Description: Store the star formation history in the output file.

Code lines: 22

Modules used: node_component_spheroid_standard galacticus_output_star_formation_-

histories iso_c_binding

kind_numbers

subroutine: node_component_spheroid_standard_star_formation_history_rate

Description: Adjust the rates for the star formation history.

Code lines: 29

Modules used: node_component_spheroid_standard galacticus_error memory_management

function: node_component_spheroid_standard_star_formation_rate

Description: Return the star formation rate of the standard spheroid.

Code lines: 24

Modules used: node_component_spheroid_standard star_formation_timescales_spheroids

function: node_component_spheroid_standard_velocity_solve

Description: Return the circular velocity of the standard spheroid.

Code lines: 9

Modules used: node_component_spheroid_standard
subroutine: node_component_spheroid_standard_velocity_solve_set
Description: Set the scale velocity of the standard spheroid.
Code lines: 10
Contained by: module node_component_spheroid_standard

file: objects.nodes.components.spheroid.standard.data.F90
Description: Contains a module of data for standard spheroid components.
Code lines: 29

module: node_component_spheroid_standard_data
Description: Contains data for standard spheroid components.
Code lines: 9
Contained by: file objects.nodes.components.spheroid.standard.data.F90
Modules used: mass_distributions
Used by: function node_component_spheroid_standard_density
        function node_component_spheroid_standard_enclosed_mass
        function node_component_spheroid_standard_half_mass_radius
        function node_component_spheroid_standard_potential
        module node_component_spheroid_standard

file: objects.nodes.components.spin.preset.F90
Description: Contains a module which implements the preset spin component.
Code lines: 159

module: node_component_spin_preset
Description: Implements the preset spin component.
Code lines: 139
Contained by: file objects.nodes.components.spin.preset.F90
Modules used: galacticus_nodes
Used by: subroutine tree_node_compute_derivatives
         subroutine tree_node_promote
         subroutine tree_node_evolve
         subroutine merger_tree_initialize

subroutine: node_component_spin_preset_initialize
Description: Initialize the spin of thisNode.
Code lines: 30
Contained by: module node_component_spin_preset

subroutine: node_component_spin_preset_promote
Description: Ensure that thisNode is ready for promotion to its parent. In this case, we simply update
the spin of thisNode to be that of its parent.
Code lines: 24
Contained by: module node_component_spin_preset
Modules used: galacticus_error

subroutine: node_component_spin_preset_rate_compute
Description: Compute rates of change of properties in the preset implementation of the spin component.
19. Source Code Documentation

**Code lines:** 17  
**Contained by:** module `node_component_spin_preset`

**subroutine:** `node_component_spin_preset_scale_set`  
**Description:** Set scales for properties in the preset implementation of the spin component.  
**Code lines:** 17  
**Contained by:** module `node_component_spin_preset`

**file:** `objects.nodes.components.spin.preset3D.F90`  
**Description:** Contains a module which implements the preset 3-D spin component.  
**Code lines:** 164

**module:** `node_component_spin_preset3d`  
**Description:** Implements the preset spin component.  
**Code lines:** 144  
**Contained by:** file `objects.nodes.components.spin.preset3D.F90`  
**Modules used:** galacticus_nodes  
**Used by:** subroutine `tree_node_compute_--derivatives`  
**subroutine:** `node_component_spin_preset3d_promote`  
**Description:** Ensure that `thisNode` is ready for promotion to its parent. In this case, we simply update the spin of `thisNode` to be that of its parent.  
**Code lines:** 24  
**Contained by:** module `node_component_spin_preset3d`  
**Modules used:** galacticus_error

**subroutine:** `node_component_spin_preset3d_rate_compute`  
**Description:** Compute rates of change of properties in the preset implementation of the spin component.  
**Code lines:** 17  
**Contained by:** module `node_component_spin_preset3d`

**module:** `node_component_spin_random`  
**Description:** Implement random spin tree node method.  
**Code lines:** 165
19.1. Program units

Code lines: 145
Contained by: file objects.nodes.components.spin.random.F90
Modules used: galacticus_nodes
Used by: subroutine tree_node_promote subroutine merger_tree_initialize

**subroutine: node_component_spin_random_initialize**

*Description:* Initializes the random spin component module.

*Code lines:* 25
*Contained by:* module node_component_spin_random
*Modules used:* galacticus_nodes

**subroutine: node_component_spin_random_initialize_spins**

*Description:* Initialize the spin of thisNode.

*Code lines:* 45
*Contained by:* module node_component_spin_random
*Modules used:* halo_spin_distributions

**subroutine: node_component_spin_random_promote**

*Description:* Ensure that thisNode is ready for promotion to its parent. In this case, we simply update the spin of thisNode to be that of its parent.

*Code lines:* 23
*Contained by:* module node_component_spin_random
*Modules used:* galacticus_error

**file: objects.radiation.F90**

*Description:* Contains a module which defines the radiation structure data type, used to describe radiation fields. (Currently only includes CMB radiation temperature.)

*Code lines:* 397

**module: radiation_structure**

*Description:* Defines the radiation structure data type, used to describe radiation fields. (Currently only includes CMB radiation temperature.)

*Code lines:* 376
*Contained by:* file objects.radiation.F90
*Used by:*

subroutine chemical_reaction_rate

subroutine chemical_hydrogen_rate_h2_.gamma_to_2h

subroutine chemical_hydrogen_rate_h2_.gamma_to_h2star_to_2h

subroutine chemical_hydrogen_rate_h2_.hplus_to_h2plus_.h

subroutine chemical_hydrogen_rate_.h2plus_.gamma_to_2hplus_.electron

subroutine chemical_hydrogen_rate_.h2plus_.hplus_to_h2plus_.h

subroutine chemical_hydrogen_rate_.h2plus_.hminus_to_h2.h

subroutine chemical_hydrogen_rate_.electron_to_h.minus photon

subroutine chemical_hydrogen_rate_.electron_to_hplus.2electron

subroutine chemical_hydrogen_rate_.h2_.hplus.to.h2plus_.h

subroutine chemical_hydrogen_rate_.h2_.hminus_to_hplus_.h

subroutine chemical_hydrogen_rate_.h2_.hplus_to_h2plus_.h

subroutine chemical_hydrogen_rate_.h2_.hminus_to_h2.h
subroutine chemical_hydrogen_rate_h-_gamma_to_hplus_electron  subroutine chemical_hydrogen_rate_h-_hplus_to_h2plus_photon  subroutine chemical_hydrogen_rate_h-_hminus_gamma_to_h_electron  subroutine chemical_hydrogen_rate_h-_hminus_to_h2_electron  subroutine chemical_hydrogen_rate_h-_hminus_electron_to_h_2electron  subroutine chemical_hydrogen_rate_h-_hminus_hplus_to_2h_electron  subroutine chemical_hydrogen_rate_h-_hminus_hplus_to_h2plus_electron  subroutine chemical_hydrogen_rates_compute
subroutine chemical_densities_cie_file_interpolate  function electron_density_cie_file_interpolate  function electron_density_cie_file_logtemperature_interpolate  function electron_density_cie_file_log_slope_cie_file  function electron_density_density_log_slope_atomic_cie_cloudy  function electron_density_temperature_log_slope_atomic_cie_cloudy  subroutine cooling_function_density_slope_molecular_hydrogen_gp  subroutine cooling_function_molecular_hydrogen_gp  module cooling_radii_isothermal  module cooling_radii_simple  module cooling_times  function cooling_time_density_log_slope_simple  function cooling_time_temperature_log_slope_simple
### 19.1. Program units

**function** `intergalactic_medium_state_`  
**module** `node_component_hot_halo_cold_`  
**internal_odes**  
**module** `node_component_hot_halo_`  
**standard**

**function**: `cross_section_integrand`  
**Description**: Integrand function used in integrating a radiation field over a cross section function.  
**Code lines**: 13  
**Contained by**: module `radiation_structure`  
**Modules used**: `iso_c_binding`

**subroutine**: `radiation_define`  
**Description**: Define which radiation fields are active in this radiation object.  
**Code lines**: 20  
**Contained by**: module `radiation_structure`  
**Modules used**: `memory_management`

**function**: `radiation_flux`  
**Description**: Return the flux of the radiation object in units of ergs cm$^{-2}$ Hz$^{-1}$ ster$^{-1}$ at the specified wavelength (in Å).  
**Code lines**: 42  
**Contained by**: module `radiation_structure`  
**Modules used**: `radiation_cmb`  
`radiation_intergalactic_background`  
`radiation_null`

**function**: `radiation_integrate_over_cross_section`  
**Description**: Integrate the photon number of the radiation field over a given cross-section function (which should return the cross section in units of cm$^2$), i.e.:  
\[
\frac{4\pi}{h} \int_{\lambda_1}^{\lambda_2} \sigma(\lambda) j_\nu(\lambda) \frac{d\lambda}{\lambda},
\]  
(19.22)

where $j_\nu$ is the flux of energy per unit area per unit solid angle and per unit frequency.  
**Code lines**: 36  
**Contained by**: module `radiation_structure`  
**Modules used**: `iso_c_binding`  
`numerical_constants_physical`  
`numerical_constants_units`  
`numerical_integration`

**function**: `radiation_is_defined`  
**Description**: Return true if the radiation object has been defined, false otherwise.  
**Code lines**: 7  
**Contained by**: module `radiation_structure`

**subroutine**: `radiation_set_node`  
**Description**: Set the radiation field as specified.  
**Code lines**: 48  
**Contained by**: module `radiation_structure`  
**Modules used**: `galacticus_nodes`  
`radiation_cmb`  
`radiation_intergalactic_background`  
`radiation_null`
**Source Code Documentation**

**subroutine: radiation_set_time**
*Description:* Set the radiation field as specified.
*Code lines:* 46
*Contained by:* module radiation_structure
*Modules used:* galacticus_nodes radiation_cmb
radioactive_intergalactic_background radiation_null

**function: radiation_temperature**
*Description:* Return the temperature of the radiation object.
*Code lines:* 40
*Contained by:* module radiation_structure
*Modules used:* radiation_cmb
radioactive_intergalactic_background radiation_null

**function: radiation_time**
*Description:* Return the time of the radiation object.
*Code lines:* 7
*Contained by:* module radiation_structure

**type: radiationdata**
*Description:* A structure used to store data for components of radiation objects.
*Code lines:* 3
*Contained by:* module radiation_structure

**type: radiationstructure**
*Description:* The radiation structure data type, used to describe radiation fields.
*Code lines:* 64
*Contained by:* module radiation_structure

**file: objects.stellar_luminosities.F90**
*Description:* Contains a module which defines the stellar luminosities object.
*Code lines:* 1074

**module: stellar_luminosities_structure**
*Description:* Defines the stellar luminosities object.
*Code lines:* 1054
*Contained by:* file objects.stellar_luminosities.F90
*Modules used:* iso_varying_string
*Used by:* subroutine galacticus_merger_tree_output_filter_luminosity
subroutine galacticus_merger_tree_output_filter_luminosity_initialize
subroutine galacticus_merger_tree_half_light
subroutine galacticus_merger_tree_half_light_initialize
subroutine galacticus_merger_tree_half_light_names
subroutine galacticus_merger_tree_half_light_property_count
subroutine galacticus_merger_tree_half_mass
subroutine galacticus_merger_tree_half_mass_initialize

1724
subroutine galacticus_output_tree_half_mass_names
subroutine galacticus_output_tree_half_mass_property_count
subroutine galacticus_output_tree_rotation_curve_initialize
module galacticus_nodes
subroutine node_component_disk_exponential_initialize
subroutine node_component_disk_exponential_post_evolve
subroutine node_component_disk_exponential_rate_compute
subroutine node_component_disk_exponential_satellite_merging
subroutine node_component_disk_exponential_scale_set
subroutine node_component_spheroid_standard_initialize
subroutine node_component_spheroid_standard_post_evolve
subroutine node_component_spheroid_standard_rate_compute
subroutine node_component_spheroid_standard_satellite_merging
subroutine node_component_spheroid_standard_scale_set
subroutine stellar_population_properties_rates
subroutine stellar_population_properties_rates_instantaneous
subroutine stellar_population_properties_rates_noninstantaneous

interface: max
Code lines: 2
Contained by: module stellar_luminosities_structure

interface: operator(*)
Code lines: 2
Contained by: module stellar_luminosities_structure

function: stellar_luminosities_add
Description: Add two stellar luminosities objects.
Code lines: 17
Contained by: module stellar_luminosities_structure

subroutine: stellar_luminosities_builder
Description: Build a stellarLuminosities object from the given XML stellarLuminositiesDefinition.
Code lines: 20
Contained by: module stellar_luminosities_structure
Modules used: fox_dom galacticus_error

subroutine: stellar_luminosities_create
Description: Ensure that the luminosity array in a stellarLuminosities is allocated.
Code lines: 8
Contained by: module stellar_luminosities_structure
Modules used: memory_management

subroutine: stellar_luminosities_deserialize
Description: Pack stellar luminosities from an array into a stellarLuminosities structure.
Code lines: 16
Contained by: module stellar_luminosities_structure
subroutine: stellar_luminosities_destroy
Description: Destroy an stellarLuminosities object.
Code lines: 8
Contained by: module stellar_luminosities_structure
Modules used: memory_management

subroutine: stellar_luminosities_destructor
Description: Destructor for a stellarLuminosities object.
Code lines: 8
Contained by: module stellar_luminosities_structure
Modules used: memory_management

function: stellar_luminosities_divide
Description: Divide a stellar luminosities object by a scalar.
Code lines: 11
Contained by: module stellar_luminosities_structure

subroutine: stellar_luminosities_dump
Description: Dump a stellar luminosities object.
Code lines: 20
Contained by: module stellar_luminosities_structure
Modules used: galacticus_display

subroutine: stellar_luminosities_dump_raw
Description: Dump an stellarLuminosities object to binary.
Code lines: 11
Contained by: module stellar_luminosities_structure

subroutine: stellar_luminosities-expand_filter_set
Description: Expand the filter set by removing the filter at index expandFrom by adding expandCount replicas of the filter at that point.
Code lines: 48
Contained by: module stellar_luminosities_structure
Modules used: iso_c_binding memory_management

subroutine: stellar_luminosities_increment
Description: Increment a stellar luminosities object.
Code lines: 11
Contained by: module stellar_luminosities_structure

function: stellar_luminosities_index
Description: Return the index of and specified entry in the luminosity list given its name.
Code lines: 15
Contained by: module stellar_luminosities_structure
Modules used: galacticus_error

subroutine: stellar_luminosities_initialize
Description: Initialize the stellarLuminositiestructure object module. Determines which stellar luminosities are to be tracked.
19.1. Program units

Code lines: 195
Contained by: module stellar_luminosities_structure
Modules used: cosmology_functions
input_parameters
memory_management
galacticus_error
instruments_filters
stellar_population_spectra_postprocess

function: stellar_luminosities_is_output
Description: Return true or false depending on whether luminosityIndex should be output at time.
Code lines: 21
Contained by: module stellar_luminosities_structure
Modules used: galacticus_error

function: stellar_luminosities_is_zero
Description: Test whether an stellarLuminosities object is zero.
Code lines: 13
Contained by: module stellar_luminosities_structure

function: stellar_luminosities_luminosity
Description: Return the requested luminosity from a stellarLuminosities object.
Code lines: 20
Contained by: module stellar_luminosities_structure
Modules used: galacticus_error

function: stellar_luminosities_multiply
Description: Multiply a stellar luminosities object by a scalar.
Code lines: 11
Contained by: module stellar_luminosities_structure

function: stellar_luminosities_multiply_switched
Description: Multiply a stellar luminosities object by a scalar.
Code lines: 11
Contained by: module stellar_luminosities_structure

function: stellar_luminosities_name
Description: Return a name for the specified entry in the stellar luminosities structure.
Code lines: 16
Contained by: module stellar_luminosities_structure
Modules used: galacticus_error

subroutine: stellar_luminosities_output
Description: Store a stellarLuminosities object in the output buffers.
Code lines: 22
Contained by: module stellar_luminosities_structure
Modules used: kind_numbers

subroutine: stellar_luminosities_output_count
Description: Increment the output count to account for a stellarLuminosities object.
Code lines: 11
function: stellar_luminosities_output_count_get
Description: Compute the number of luminosities to be output at a given time.
Code lines: 13
Contained by: module stellar_luminosities_structure

subroutine: stellar_luminosities_output_names
Description: Assign names to output buffers for a stellarLuminosities object.
Code lines: 25
Contained by: module stellar_luminosities_structure

function: stellar_luminosities_property_count
Description: Return the number of properties required to track stellar luminosities.
Code lines: 8
Contained by: module stellar_luminosities_structure

subroutine: stellar_luminosities_read_raw
Description: Read an stellarLuminosities object from binary.
Code lines: 14
Contained by: module stellar_luminosities_structure

subroutine: stellar_luminosities_reset
Description: Reset an stellarLuminosities object.
Code lines: 12
Contained by: module stellar_luminosities_structure

subroutine: stellar_luminosities_serialize
Description: Unpack stellar luminosities from a stellarLuminosities structure into an array.
Code lines: 15
Contained by: module stellar_luminosities_structure

subroutine: stellar_luminosities_set
Description: Set the luminosity in each band for a single stellar population of given mass with the specified abundancesStellar and which formed at cosmological time with IMF specified by imfSelected.
Code lines: 24
Contained by: module stellar_luminosities_structure
Modules used: abundances_structure stellar_population_luminosities

subroutine: stellar_luminosities_set_to_unity
Description: Set an stellarLuminosities object to unity.
Code lines: 12
Contained by: module stellar_luminosities_structure

subroutine: stellar_luminosities_special_cases
Description: Modify the input list of luminosities for special cases.
Code lines: 97
Contained by: module stellar_luminosities_structure
19.1. Program units

Modules used:

- cosmology_functions
- galacticus_output_times
- iso_c_binding
- memory_management
- string_handling

Function: stellar_luminosities_subtract
Description: Subtract two stellar luminosities objects.
Code lines: 17
Contained by: module stellar_luminosities_structure

Type: stellar_luminosities
Description: The stellar luminosities structure.
Code lines: 195
Contained by: module stellar_luminosities_structure

Function: stellar_luminosities_max
Description: Return an element-by-element max() on two stellar luminosity objects.
Code lines: 8
Contained by: module stellar_luminosities_structure

File: objects.tables.F90
Description: Contains a module which defines a table class with optimized interpolation operators.
Code lines: 1053

Module: tables
Description: Defines a table class with optimized interpolation operators.
Code lines: 1033
Contained by: file objects.tables.F90
Modules used: fgs1

Used by:
- module accretion_disks_adaf
- file dark_matter_halos.scales.virial_density_contrast.F90
- file dark_matter_profiles.Einasto.F90
- file dark_matter_profiles.NFW.F90
- module conditional_mass_functions_behroozi2010
- file hot_halo.mass.distribution.cored.core_radius.growing.F90
- function node_component_disk_exponential_density
- function node_component_disk_exponential_potential
- function node_component_disk_exponential_rotation_curve
- function node_component_disk_exponential_rotation_curve_gradient
- subroutine node_component_disk_exponential_initialize

- module dark_matter_halo_scales
- module halo_spin_distributions_bett2007
- module dark_matter_profiles
- module galacticus_state
- module hot_halo_mass_distributions_core_radii
- module intergalactic_medium_state
- function node_component_disk_exponential_enclosed_mass
- function node_component_disk_exponential_surface_density
- subroutine node_component_disk_exponential_radius_solver

1729
module node_component_disk_exponential_data
subroutine star_formation_imf_tabulate_baugh2005toheavey
subroutine imf_tabulate
subroutine star_formation_imf_tabulate_kroupa
subroutine star_formation_imf_tabulate_salpeter
subroutine star_formation_imf_tabulate_piecewisepowerlaw
subroutine star_formation_imf_tabulate_kennicutt
subroutine star_formation_imf_tabulate_millerscalo
subroutine star_formation_imf_tabulate_salpeter
subroutine star_formation_imf_tabulate_chabrier
subroutine star_formation_imf_tabulate_scalo
subroutine star_formation_imf_tabulate_salpeter

module virial_orbits_wetzel2010
subroutine star_formation_imf_tabulate_kroupa
subroutine star_formation_imf_tabulate_salpeter
subroutine star_formation_imf_tabulate_kennicutt
subroutine star_formation_imf_tabulate_millerscalo
subroutine star_formation_imf_tabulate_salpeter
subroutine star_formation_imf_tabulate_chabrier
subroutine star_formation_imf_tabulate_scalo
subroutine star_formation_imf_tabulate_salpeter

module critical_overdensity
subroutine critical_overdensity_kitayama_suto1996
subroutine make_table
subroutine spherical_collape_matter_lambda_delta_virial_tabulate
subroutine stellar_population_spectra_conroy_initialize_imf

module critical_overdensity
subroutine make_table
subroutine spherical_collape_matter_lambda_delta_virial_tabulate
subroutine make_table

subroutine cosmological_mass_variance_filtered_power_spectrum_tabulate
subroutine critical_overdensity_kitayama_suto1996
subroutine make_table
subroutine spherical_collape_matter_lambda_delta_virial_tabulate
subroutine make_table

subroutine spherical_collape_matter_lambda_delta_virial_tabulate
subroutine make_table

file structure_formation.virial_density_contrast.spherical_collape_matter_lambda.F90

---

type: table
Description: Basic table type.
Code lines: 13
Contained by: module tables

---

type: table1d
Description: Basic table type.
Code lines: 80
Contained by: module tables

---

function: table1d_find_effective_x
Description: Return the effective value of x to use in table interpolations.
Code lines: 24
Contained by: module tables
Modules used: galacticus_error

---

function: table1d_is_monotonic
Description: Return true if a 1D table is monotonic. Optionally allows specification of the direction, and whether or not equal elements are allowed for monotonicity.
Code lines: 14
19.1. Program units

**Contained by:** module `tables`

**Modules used:** `array_utilities`

**function: table1d_size**
- **Description:** Return the size of a 1D table.
- **Code lines:** 7
- **Contained by:** module `tables`

**function: table1d_x**
- **Description:** Return the \(i^{th}\) x-value for a 1D table.
- **Code lines:** 11
- **Contained by:** module `tables`

**function: table1d_xs**
- **Description:** Return the x-values for a 1D table.
- **Code lines:** 8
- **Contained by:** module `tables`

**function: table1d_y**
- **Description:** Return the \(i^{th}\) y-value for a 1D table.
- **Code lines:** 14
- **Contained by:** module `tables`

**function: table1d_ys**
- **Description:** Return the y-values for a 1D table.
- **Code lines:** 8
- **Contained by:** module `tables`

**type: table1dgeneric**
- **Description:** Table type supporting generic one dimensional tables.
- **Code lines:** 28
- **Contained by:** module `tables`

**type: table1dlinearcspline**
- **Description:** Table type supporting one dimensional table with linear spacing in \(x\) and cubic spline interpolation.
- **Code lines:** 29
- **Contained by:** module `tables`

**type: table1dlinearlinear**
- **Description:** Table type supporting one dimensional table with linear spacing in \(x\).
- **Code lines:** 26
- **Contained by:** module `tables`

**type: table1dlogarithmicspline**
- **Description:** Table type supporting one dimensional table with logarithmic spacing in \(x\) and cubic spline interpolation.
- **Code lines:** 11
- **Contained by:** module `tables`
type: table1dlogarithmiclinear
Description: Table type supporting one dimensional table with logarithmic spacing in x.
Code lines: 10
Contained by: module tables

subroutine: table_1d_destroy
Description: Destroy a 1-D table.
Code lines: 9
Contained by: module tables
Modules used: memory_management

subroutine: table_1d_reverse
Description: Reverse a 1D table (i.e. swap x and y components). Optionally allows specification of which y table to swap with.
Code lines: 34
Contained by: module tables
Modules used: array_utilities galacticus_error

subroutine: table_generic_1d_create
Description: Create a 1-D generic table.
Code lines: 25
Contained by: module tables
Modules used: memory_management

subroutine: table_generic_1d_destroy
Description: Destroy a generic 1-D table.
Code lines: 9
Contained by: module tables
Modules used: numerical_interpolation

function: table_generic_1d_interpolate
Description: Perform generic interpolation in a generic 1D table.
Code lines: 13
Contained by: module tables
Modules used: numerical_interpolation

function: table_generic_1d_interpolate_gradient
Description: Perform generic interpolation in a generic 1D table.
Code lines: 13
Contained by: module tables
Modules used: numerical_interpolation

subroutine: table_generic_1d_populate
Description: Populate a 1-D generic table.
Code lines: 20
Contained by: module tables
Modules used: galacticus_error
19.1. Program units

**subroutine: table_generic_1d_populate_single**
*Description:* Populate a single element of a 1-D generic table.
*Code lines:* 21
*Contained by:* module `tables`
*Modules used:* `galacticus_error`

**subroutine: table_linear_1d_create**
*Description:* Create a 1-D linear table.
*Code lines:* 29
*Contained by:* module `tables`
*Modules used:* `galacticus_error`, `memory_management`, `numerical_ranges`

**function: table_linear_1d_interpolate**
*Description:* Perform linear interpolation in a linear 1D table.
*Code lines:* 33
*Contained by:* module `tables`

**function: table_linear_1d_interpolate_gradient**
*Description:* Perform linear interpolation in a linear 1D table.
*Code lines:* 30
*Contained by:* module `tables`

**subroutine: table_linear_1d_populate**
*Description:* Populate a 1-D linear table.
*Code lines:* 24
*Contained by:* module `tables`
*Modules used:* `galacticus_error`

**subroutine: table_linear_1d_populate_single**
*Description:* Populate a single element of a 1-D linear table.
*Code lines:* 25
*Contained by:* module `tables`
*Modules used:* `galacticus_error`

**subroutine: table_linear_cspline_1d_coefficients**
*Description:* Compute coefficients for a spline interpolation.
*Code lines:* 21
*Contained by:* module `tables`

**subroutine: table_linear_cspline_1d_compute_spline**
*Description:* Compute the interpolating spline factors for a 1-D linear spline.
*Code lines:* 34
*Contained by:* module `tables`

**subroutine: table_linear_cspline_1d_create**
*Description:* Create a 1-D linear table.
*Code lines:* 30
*Contained by:* module `tables`
19. Source Code Documentation

**subroutine:** table_linear_cspline_1d_destroy  
*Description:* Destroy a linear cubic-spline 1-D table.  
*Code lines:* 9  
*Contained by:* module tables  
*Modules used:* memory_management

**function:** table_linear_cspline_1d_interpolate  
*Description:* Perform linear interpolation in a linear 1D table.  
*Code lines:* 32  
*Contained by:* module tables

**function:** table_linear_cspline_1d_interpolate_gradient  
*Description:* Perform linear interpolation in a linear 1D table.  
*Code lines:* 32  
*Contained by:* module tables

**subroutine:** table_linear_cspline_1d_populate  
*Description:* Populate a 1-D linear table.  
*Code lines:* 27  
*Contained by:* module tables  
*Modules used:* galacticus_error

**subroutine:** table_linear_cspline_1d_populate_single  
*Description:* Populate a single element of a 1-D linear table.  
*Code lines:* 28  
*Contained by:* module tables  
*Modules used:* galacticus_error

**subroutine:** table_logarithmic_1d_create  
*Description:* Create a 1-D logarithmic table.  
*Code lines:* 12  
*Contained by:* module tables

**function:** table_logarithmic_1d_interpolate  
*Description:* Perform linear interpolation in a logarithmic 1D table.  
*Code lines:* 14  
*Contained by:* module tables

**function:** table_logarithmic_1d_interpolate_gradient  
*Description:* Perform linear interpolation in a logarithmic 1D table.  
*Code lines:* 14  
*Contained by:* module tables

**function:** table_logarithmic_1d_x  
*Description:* Return the $i^{th}$ $x$-value for a logarithmic 1D table.  
*Code lines:* 8  
*Contained by:* module tables
function: table_logarithmic_1d_xs
Description: Return the $x$-values for a 1D table.
Code lines: 8
Contained by: module tables

subroutine: table_logarithmic_cspline_1d_create
Description: Create a 1-D logarithmic table.
Code lines: 15
Contained by: module tables

function: table_logarithmic_cspline_1d_interpolate
Description: Perform linear interpolation in a logarithmic 1D table.
Code lines: 14
Contained by: module tables

function: table_logarithmic_cspline_1d_interpolate_gradient
Description: Perform linear interpolation in a logarithmic 1D table.
Code lines: 14
Contained by: module tables

function: table_logarithmic_cspline_1d_x
Description: Return the $i^{th}$ $x$-value for a logarithmic 1D table.
Code lines: 16
Contained by: module tables

function: table_logarithmic_cspline_1d_xs
Description: Return the $x$-values for a 1D table.
Code lines: 8
Contained by: module tables

file: objects.tensors.F90
Description: Contains a module which defines the tensor symmetric structure used for describing symmetric tensors.
Code lines: 647

module: tensors
Description: Defines the tensor symmetric structure used for describing symmetric tensors.
Code lines: 625
Contained by: file objects.tensors.F90
Modules used: iso_varying_string numerical_constants_astronomical
Used by: function vector_self_outer_product module galacticus_nodes
module node_component_satellite_orbiting function satellite_tidal_heating_rate_gnedin
program test_tensors

interface: assignment(=)
Code lines: 3
Contained by: module tensors
interface: max
  Code lines: 2
  Contained by: module tensors

interface: operator(*)
  Code lines: 2
  Contained by: module tensors

type: tensor
  Description: A generic tensor type.
  Code lines: 2
  Contained by: module tensors

function: tensor_r2_d3_sym_add
  Description: Add two tensorRank2Dimension3Symmetric objects.
  Code lines: 22
  Contained by: module tensors

subroutine: tensor_r2_d3_sym_assign_from
  Description: Assign a matrix to a tensorRank2Dimension3Symmetric object.
  Code lines: 16
  Contained by: module tensors

subroutine: tensor_r2_d3_sym_assign_to
  Description: Assign a matrix to a tensorRank2Dimension3Symmetric object.
  Code lines: 9
  Contained by: module tensors
  Modules used: galacticus_error

subroutine: tensor_r2_d3_sym_builder
  Description: Build a tensorRank2Dimension3Symmetric object from the given XML tensorDefinition.
  Code lines: 34
  Contained by: module tensors
  Modules used: fox_dom galacticus_error

function: tensor_r2_d3_sym_contract
  Description: Return the contraction (trace) of a tensorRank2Dimension3Symmetric object.
  Code lines: 7
  Contained by: module tensors

subroutine: tensor_r2_d3_sym_deserialize
  Description: Pack an array into a tensorRank2Dimension3Symmetric symmetric structure.
  Code lines: 13
  Contained by: module tensors

subroutine: tensor_r2_d3_sym_destroy
  Description: Destroy a tensorRank2Dimension3Symmetric symmetric object.
  Code lines: 7
19.1. Program units

Contained by: module tensors
Modules used: memory_management

function: tensor_r2_d3_sym_double_contract
Description: Find the double contraction of two tensorRank2Dimension3Symmetric objects, A:B.
Code lines: 7
Contained by: module tensors

subroutine: tensor_r2_d3_sym_dump
Description: Reset a tensorRank2Dimension3Symmetric symmetric object.
Code lines: 28
Contained by: module tensors
Modules used: galacticus_display iso_varying_string

subroutine: tensor_r2_d3_sym_dump_raw
Description: Dump a tensorRank2Dimension3Symmetric object to binary.
Code lines: 16
Contained by: module tensors
Modules used: galacticus_display iso_varying_string

subroutine: tensor_r2_d3_sym_from_matrix
Description: Construct a tensorRank2Dimension3Symmetric object from a matrix.
Code lines: 17
Contained by: module tensors
Modules used: galacticus_error

subroutine: tensor_r2_d3_sym_increment
Description: Increment a tensorRank2Dimension3Symmetric object.
Code lines: 13
Contained by: module tensors

function: tensor_r2_d3_sym_is_zero
Description: Test whether a tensorRank2Dimension3Symmetric object is zero.
Code lines: 8
Contained by: module tensors

function: tensor_r2_d3_sym_matrix_equality
Description: Return true if the supplied tensor and matrix are equal.
Code lines: 8
Contained by: module tensors

function: tensor_r2_d3_sym_max
Description: Return an element-by-element max() on two tensorRank2Dimension3Symmetric objects.
Code lines: 13
Contained by: module tensors

function: tensor_r2_d3_sym_property_count
Description: Return the number of properties required to track a rank 2, 3 dimensional, symmetric tensor.
   This is equal to 6.
Code lines: 7
19. Source Code Documentation

**Contained by:** module `tensors`

**subroutine: tensor_r2_d3_sym_read_raw**
*Description:* Read a `tensorRank2Dimension3Symmetric` object from binary.
*Code lines:* 16
*Contained by:* module `tensors`
*Modules used:* `galacticus_display` `iso_varying_string`

**subroutine: tensor_r2_d3_sym_reset**
*Description:* Reset a `tensorRank2Dimension3Symmetric` object.
*Code lines:* 13
*Contained by:* module `tensors`

**function: tensor_r2_d3_sym_scalar_divide**
*Description:* Multiply a `tensorRank2Dimension3Symmetric` object by a scalar.
*Code lines:* 14
*Contained by:* module `tensors`

**function: tensor_r2_d3_sym_scalar_multiply**
*Description:* Multiply a `tensorRank2Dimension3Symmetric` object by a scalar.
*Code lines:* 14
*Contained by:* module `tensors`

**function: tensor_r2_d3_sym_scalar_multiply_switched**
*Description:* Multiply a scalar by a `tensorRank2Dimension3Symmetric` object.
*Code lines:* 9
*Contained by:* module `tensors`

**subroutine: tensor_r2_d3_sym_serialize**
*Description:* Pack a `tensorRank2Dimension3Symmetric` into an array.
*Code lines:* 14
*Contained by:* module `tensors`

**subroutine: tensor_r2_d3_sym_set_to_identity**
*Description:* Set a `tensorRank2Dimension3Symmetric` object to the identity matrix.
*Code lines:* 13
*Contained by:* module `tensors`

**subroutine: tensor_r2_d3_sym_set_to Unity**
*Description:* Set a `tensorRank2Dimension3Symmetric` object to unity.
*Code lines:* 13
*Contained by:* module `tensors`

**function: tensor_r2_d3_sym_subtract**
*Description:* Subtract two `tensorRank2Dimension3Symmetric` objects.
*Code lines:* 23
*Contained by:* module `tensors`

**function: tensor_r2_d3_sym_to_matrix**
19.1. Program units

**Description:** Construct a matrix from a `tensorRank2Dimension3Symmetric`.

**Code lines:** 16

**Contained by:** module `tensors`

**type:** `tensorRank2Dimension3Symmetric`

**Description:** A rank 2, three dimensional, symmetric tensor.

**Code lines:** 167

**Contained by:** module `tensors`

**file:** `optimal_sampling.stellar_mass_function.F90`

**Description:** Contains a program which computes the optimal sampling of the halo mass function to minimize errors on the stellar mass function.

**Code lines:** 191

**program:** `optimal_sampling_smf`

**Description:** Compute the optimal number of trees to run of each mass.

**Code lines:** 170

**Contained by:** file `optimal_sampling.stellar_mass_function.F90`

**Modules used:**
- `cosmology_functions`
- `fgsl`
- `galacticus_error`
- `galacticus_meta_compute_times`
- `halo_mass_function`
- `input_parameters`
- `io_hdf5`
- `iso_c_binding`
- `iso_varying_string`
- `memory_management`
- `merger_trees_mass_function_sampling`
- `numerical_constants_astronomical`
- `numerical_integration`
- `numerical_ranges`

**function:** `stellar_mass_function_integrand`

**Description:** The integrand (as a function of halo mass) giving the stellar mass function.

**Code lines:** 14

**Contained by:** program `optimal_sampling_smf`

**Modules used:** `conditional_mass_functions`

**file:** `power_spectra.tasks.F90`

**Description:** Contains a module which implements calculations of power spectra and related properties for output.

**Code lines:** 165

**module:** `power_spectrum_tasks`

**Description:** Implements calculations of power spectra and related properties for output.

**Code lines:** 145

**Contained by:** file `power_spectra.tasks.F90`

**Modules used:** `io_hdf5`

**Used by:** program `power_spectra`

**subroutine:** `power_spectrum_close_file`

**Description:** Close the output file for power spectrum data.

**Code lines:** 6

**Contained by:** module `power_spectrum_tasks`
subroutine: power_spectrum_compute
Description: Computes power spectra and related properties for output.
Code lines: 76
Contained by: module power_spectrum_tasks
Modules used:
- cosmology_parameters
- memory_management
- numerical_ranges
- input_parameters
- numerical_constants_math
- power_spectra

subroutine: power_spectrum_open_file
Description: Open the output file for power spectrum data.
Code lines: 14
Contained by: module power_spectrum_tasks
Modules used:
- hdf5
- power_spectra_tasks
- iso_varying_string

subroutine: power_spectrum_output
Description: Outputs power spectrum data.
Code lines: 26
Contained by: module power_spectrum_tasks
Modules used:
- numerical_constants_astronomical

file: radiation.cosmic_microwave_background.F90
Description: Contains a module which implements a cosmic microwave background radiation component.
Code lines: 153

module: radiation_cmb
Description: Implements a cosmic microwave background radiation component.
Code lines: 133
Contained by: file radiation.cosmic_microwave_background.F90
Used by:
- function radiation_flux
- subroutine radiation_set_node
- subroutine radiation_set_time
- function radiation_temperature

subroutine: radiation_flux_cmb
Description: Flux method for the CMB radiation component.
Code lines: 26
Contained by: module radiation_cmb
Modules used:
- numerical_constants_units
- thermodynamics_radiation

subroutine: radiation_set_cmb
Description: Property setting routine for the cosmic microwave background radiation component.
Code lines: 25
Contained by: module radiation_cmb
Modules used:
- cosmology_functions
- galacticus_nodes
- memory_management

subroutine: radiation_set_time_cmb
Description: Property setting routine for the cosmic microwave background radiation component.
Code lines: 22
Contained by: module radiation_cmb
19.1. Program units

 Modules used:  cosmology_functions    memory_management

 subroutine:  radiation_temperature_cmb
 Description:  Returns the temperature for the cosmic microwave background radiation component.
 Code lines:  23
 Contained by:  module radiation_cmb

 file:  radiation.intergalactic_background.F90
 Description:  Contains a module which implements an intergalactic background (excluding the CMB) radiation component.
 Code lines:  206

 module:  radiation_intergalactic_background
 Description:  Implements an intergalactic background (excluding the CMB) radiation component.
 Code lines:  186
 Contained by:  file radiation.intergalactic_background.F90
 Modules used:  galacticus_nodes
 Used by:  function radiation_flux    subroutine radiation_set_node
 subroutine radiation_set_time    function radiation_temperature

 subroutine:  radiation_flux_intergalactic_background
 Description:  Flux method for the radiation component from file method.
 Code lines:  27
 Contained by:  module radiation_intergalactic_background

 subroutine:  radiation_initialize_intergalactic_background
 Description:  Initialize the intergalactic background radiation component module.
 Code lines:  54
 Contained by:  module radiation_intergalactic_background
 Modules used:  galacticus_error    input_parameters
 iso_varying_string    radiation_igb_file
 radiation_intergalactic_background_internal

 subroutine:  radiation_set_intergalactic_background
 Description:  Property setting routine for the radiation component from file method.
 Code lines:  18
 Contained by:  module radiation_intergalactic_background

 subroutine:  radiation_set_time_intergalactic_background
 Description:  Property setting routine for the radiation component from file method.
 Code lines:  16
 Contained by:  module radiation_intergalactic_background

 subroutine:  radiation_temperature_intergalactic_background
 Description:  Returns the temperature for the radiation component from file method.
 Code lines:  10
 Contained by:  module radiation_intergalactic_background
file: radiation.intergalactic_background.file.F90
Description: Contains a module which implements an intergalactic background radiation component read from a file.
Code lines: 201

module: radiation_igb_file
Description: Implements an intergalactic background radiation component read from a file.
Code lines: 181
Contained by: file radiation.intergalactic_background.file.F90
Modules used: fgsl
Used by: subroutine radiation_initialize_intergalactic_background

subroutine: radiation_igb_file_flux
Description: Flux method for the radiation component from file method.
Code lines: 37
Contained by: module radiation_igb_file
Modules used: iso_c_binding numerical_interpolation

function: radiation_igb_file_format_version
Description: Return the current file format version of intergalactic background radiation files.
Code lines: 6
Contained by: module radiation_igb_file

subroutine: radiation_igb_file_initialize
Description: Initialize the intergalactic background radiation component from file module by reading in the data.
Code lines: 89
Contained by: module radiation_igb_file
Modules used: array_utilities cosmology_functions
fox_dom galacticus_error
galacticus_input_paths input_parameters
io_xml iso_varying_string
memory_management

subroutine: radiation_igb_file_set
Description: Property setting routine for the radiation component from file method.
Code lines: 14
Contained by: module radiation_igb_file
Modules used: galacticus_nodes memory_management

file: radiation.intergalactic_background.internal.F90
Description: Contains a module which handles evolving the spectrum of background radiation.
Code lines: 488

module: radiation.intergalactic_background_internal
Description: Handles evolving the spectrum of background radiation.
Code lines: 468
19.1. Program units

| Contained by: | file radiation.intergalactic_background.internal.F90 |
| Modules used: | abundances_structure cosmology_functions |
|              | cosmology_parameters fgs1 |
|              | intergalactic_medium_state |
| Used by:     | function galacticus_task_evolve_tree |
|              | subroutine radiation_initialize_intergalactic_background |

**function: backgroundradiationodes**

*Description:* Evaluates the ODEs controlling the evolution of cosmic background radiation.

*Code lines:* 35

**Contained by:** module radiation_intergalactic_background_internal

**Modules used:**
- iso_c_binding
- numerical_constants_atomic
- numerical_constants_units
- galacticus_error

**subroutine: radiation_igb_internal_flux**

*Description:* Flux method for the radiation component from file method.

*Code lines:* 41

**Contained by:** module radiation_intergalactic_background_internal

**Modules used:**
- fgs1
- iso_c_binding
- numerical_constants_atomic
- numerical_constants_units
- galacticus_error

**subroutine: radiation_igb_internal_initialize**

*Description:* Initialize the internally-computed intergalactic background radiation component module.

*Code lines:* 12

**Contained by:** module radiation_intergalactic_background_internal

**subroutine: radiation_igb_internal_set**

*Description:* Property setting routine for the internally-computed intergalactic background radiation component method.

*Code lines:* 12

**Contained by:** module radiation_intergalactic_background_internal

**Modules used:** memory_management

**subroutine: radiation_intergalactic_background_internal_initialize**

*Description:* Attach an initial event to a merger tree to cause the background radiation update function to be called.

*Code lines:* 134

**Contained by:** module radiation_intergalactic_background_internal

**Modules used:**
- atomic_cross_sections_ionization_photo cosmology_functions
- galacticus_nodes input_parameters
- memory_management numerical_ranges

**function: radiation_intergalactic_background_internal_update**

*Description:* Update the radiation background for a given universe.

*Code lines:* 164
function: stellarspectraconvolution
Description: Integrand for convolution of stellar spectra.
Code lines: 14
Contained by: module radiation_intergalactic_background_internal
Modules used: galacticus_error iso_c_binding
stellar_population_spectra

file: radiation.null.F90
Code lines: 89

module: radiation_null
Code lines: 71
Contained by: file radiation.null.F90
Used by: function radiation_flux subroutine radiation_set_node subroutine radiation_set_time function radiation_temperature

subroutine: radiation_flux_null
Description: Flux method for the null radiation component.
Code lines: 10
Contained by: module radiation_null

subroutine: radiation_set_null
Description: Property setting routine for null radiation component.
Code lines: 9
Contained by: module radiation_null
Modules used: galacticus_nodes

subroutine: radiation_set_time_null
Description: Property setting routine for null radiation component.
Code lines: 8
Contained by: module radiation_null

subroutine: radiation_temperature_null
Description: Temperature method for the null radiation component.
Code lines: 9
Contained by: module radiation_null
19.1. Program units

file: `ram_pressure_stripping.mass_loss_rate.disks.F90`
- **Description:** Contains a module that implements calculations of mass loss rates from disks due to ram pressure stripping.
- **Code lines:** 99

module: `ram_pressure_stripping_mass_loss_rate_disks`
- **Description:** Implements calculations of mass loss rates from disks due to ram pressure stripping.
- **Code lines:** 78
- **Contained by:** file `ram_pressure_stripping.mass_loss_rate.disks.F90`
- **Used by:** subroutine `node_component_disk_exponential_rate_compute`

function: `ram_pressure_stripping_mass_loss_rate_disk`
- **Description:** Return the ram pressure force for the hot halo of thisNode.
- **Code lines:** 62
- **Contained by:** module `ram_pressure_stripping_mass_loss_rate_disks`
- **Modules used:** `galacticus_error` `galacticus_nodes` `input_parameters` `iso_varying_string` `ram_pressure_stripping_mass_loss_rate_disks_null` `ram_pressure_stripping_mass_loss_rate_disks_simple`

file: `ram_pressure_stripping.mass_loss_rate.disks.null.F90`
- **Description:** Contains a module which implements a null mass loss rates from disks due to ram pressure stripping.
- **Code lines:** 52

module: `ram_pressure_stripping_mass_loss_rate_disks_null`
- **Description:** Implements a null mass loss rates from disks due to ram pressure stripping.
- **Code lines:** 32
- **Contained by:** file `ram_pressure_stripping.mass_loss_rate.disks.null.F90`
- **Used by:** function `ram_pressure_stripping_mass_loss_rate_disk`

function: `ram_pressure_stripping_mass_loss_rate_disk_null`
- **Description:** Computes the mass loss rate from disks due to ram pressure stripping. Always returns zero.
- **Code lines:** 8
- **Contained by:** module `ram_pressure_stripping_mass_loss_rate_disks_null`
- **Modules used:** `galacticus_nodes`

subroutine: `ram_pressure_stripping_mass_loss_rate_disks_null_init`
- **Description:** Initializes the “null” ram pressure stripping mass loss rate from disks module.
- **Code lines:** 9
- **Contained by:** module `ram_pressure_stripping_mass_loss_rate_disks_null`
- **Modules used:** `iso_varying_string`

file: `ram_pressure_stripping.mass_loss_rate.disks.simple.F90`
- **Description:** Contains a module which implements simple mass loss rates from disks due to ram pressure stripping.
- **Code lines:** 115
module: *ram_pressure_stripping_mass_loss_rate_disks_simple*

*Description:* Implements simple mass loss rates from disks due to ram pressure stripping.

*Code lines:* 95

*Contained by:* file *ram_pressure_stripping_mass_loss_rate.disks.simple.F90*

*Used by:* function *ram_pressure_stripping_mass_loss_rate_disk*

function: *ram_pressure_stripping_mass_loss_rate_disk_simple*

*Description:* Computes the mass loss rate from disks due to ram pressure stripping assuming a simple model. Specifically, the mass loss rate is

\[
\dot{M}_{\text{gas}} = -\alpha M_{\text{gas}}/\tau_{\text{disk}},
\]

where
\[
\alpha = F_{\text{ram}}/F_{\text{gravity}},
\]

\(F_{\text{ram}}\) is the ram pressure force from the hot halo (see §13.18), and

\[
F_{\text{gravity}} = 2\pi G\Sigma_{\text{gas}}(r_{1/2})\Sigma_{\text{total}}(r_{1/2})
\]

is the gravitational restoring force in the disk at the half-mass radius, \(r_{1/2}\).

*Code lines:* 54

*Contained by:* module *ram_pressure_stripping_mass_loss_rate_disks_simple*

*Modules used:* galactic_structure_options, galactic_structure_surface_densities, galacticus_nodes, hot_halo_ram_pressure_forces, numerical_constants_astronomical, numerical_constants_physical

subroutine: *ram_pressure_stripping_mass_loss_rate_disks_simple_init*

*Description:* Initializes the “simple” ram pressure stripping mass loss rate from disks module.

*Code lines:* 23

*Contained by:* module *ram_pressure_stripping_mass_loss_rate_disks_simple*

*Modules used:* input_parameters, iso_varying_string

file: *ram_pressure_stripping_mass_loss_rate.spheroids.F90*

*Description:* Contains a module that implements calculations of mass loss rates from spheroids due to ram pressure stripping.

*Code lines:* 99

module: *ram_pressure_stripping_mass_loss_rate_spheroids*

*Description:* Implements calculations of mass loss rates from spheroids due to ram pressure stripping.

*Code lines:* 78

*Contained by:* file *ram_pressure_stripping_mass_loss_rate.spheroids.F90*

*Used by:* subroutine *node_component_spheroid_standard_rate_compute*

function: *ram_pressure_stripping_mass_loss_rate_spheroid*

*Description:* Return the ram pressure force for the hot halo of thisNode.

*Code lines:* 62

*Contained by:* module *ram_pressure_stripping_mass_loss_rate_spheroids*
19.1. Program units

**Modules used:**
- galacticus_error
- galacticus_nodes
- input_parameters
- iso_varying_string
- ram_pressure_stripping_mass_loss_rate_spheroids_null
- ram_pressure_stripping_mass_loss_rate_spheroids_simple

**file:** ram_pressure_stripping.mass_loss_rate.spheroids.null.F90

*Description:* Contains a module which implements a null mass loss rates from spheroids due to ram pressure stripping.

*Code lines:* 54

**module:** ram_pressure_stripping_mass_loss_rate_spheroids_null

*Description:* Implements a null mass loss rates from spheroids due to ram pressure stripping.

*Code lines:* 34

*Contained by:* file ram_pressure_stripping.mass_loss_rate.spheroids.null.F90

*Modules used:* galacticus_nodes

*Used by:* function ram_pressure_stripping_mass_loss_rate_spheroid

**function:** ram_pressure_stripping_mass_loss_rate_spheroid_null

*Description:* Computes the mass loss rate from spheroids due to ram pressure stripping. Always returns zero.

*Code lines:* 8

*Contained by:* module ram_pressure_stripping_mass_loss_rate_spheroids_null

*Modules used:* galacticus_nodes

**subroutine:** ram_pressure_stripping_mass_loss_rate_spheroids_null_init

*Description:* Initializes the “null” ram pressure stripping mass loss rate from spheroids module.

*Code lines:* 10

*Contained by:* module ram_pressure_stripping_mass_loss_rate_spheroids_null

*Modules used:* input_parameters, iso_varying_string

**file:** ram_pressure_stripping.mass_loss_rate.spheroids.simple.F90

*Description:* Contains a module which implements simple mass loss rates from spheroids due to ram pressure stripping.

*Code lines:* 118

**module:** ram_pressure_stripping_mass_loss_rate_spheroids_simple

*Description:* Implements simple mass loss rates from spheroids due to ram pressure stripping.

*Code lines:* 98

*Contained by:* file ram_pressure_stripping.mass_loss_rate.spheroids.simple.F90

*Modules used:* galacticus_nodes

*Used by:* function ram_pressure_stripping_mass_loss_rate_spheroid

**function:** ram_pressure_stripping_mass_loss_rate_spheroid_simple
Computes the mass loss rate from spheroids due to ram pressure stripping assuming a simple model. Specifically, the mass loss rate is

\[ \dot{M}_{\text{gas}} = -\alpha M_{\text{gas}}/\tau_{\text{spheroid}}, \]  

(19.26)

where

\[ \alpha = F_{\text{ram}}/F_{\text{gravity}}, \]  

(19.27)

\( F_{\text{ram}} \) is the ram pressure force from the hot halo (see §13.18), and

\[ F_{\text{gravity}} = \frac{4}{3} \rho_{\text{gas}}(r_{1/2}) \frac{GM_{\text{total}}(r_{1/2})}{r_{1/2}} \]  

(19.28)

is the gravitational restoring force in the spheroid at the half-mass radius, \( r_{1/2} \) [Takeda et al., 1984].

---

**Code lines:** 56

**Contained by:** module `ram_pressure_stripping_mass_loss_rate_spheroids_simple`

**Modules used:** `galactic_structure_densities` `galactic_structure_enclosed_masses`
galactic_structure_options  galacticus_nodes
hot_halo_ram_pressure_forces  numerical_constants_astronomical
numerical_constants_math  numerical_constants_physical

subroutine:  ram_pressure_stripping_mass_loss_rate_spheroids_simple_init
Description:  Initializes the “simple” ram pressure stripping mass loss rate from spheroids module.
Code lines:  23
Contained by:  module  ram_pressure_stripping_mass_loss_rate_spheroids_simple
Modules used:  input_parameters  iso_varying_string

file:  satellites.dynamical_friction.acceleration.Chandrasekhar1943.F90
Description:  Contains a module with a Chandrasekhar [1943] implementation of calculations of satellite acceleration due to dynamical friction.
Code lines:  97

module:  dynamical_friction_acceleration_chandrasekhar
Description:  Implements Chandrasekhar [1943] value of calculations of satellite acceleration due to dynamical friction.
Code lines:  75
Contained by:  file  satellites.dynamical_friction.acceleration.Chandrasekhar1943.F90
Used by:  subroutine  satellite_dynamical_friction_acceleration_initialize

function:  satellite_dynamical_friction_acceleration_chandrasekhar
Description:  Return the Chandrasekhar [1943] acceleration for satellites due to dynamical friction.
Code lines:  33
Contained by:  module  dynamical_friction_acceleration_chandrasekhar
Modules used:  dark_matter_halo_scales  error_functions
galactic_structure_densities  galactic_structure_options
galacticus_nodes  numerical_constants_astronomical
numerical_constants_math  numerical_constants_physical
numerical_constants_prefixes  vectors

subroutine:  satellite_dynamical_friction_chandrasekhar_initialize
Description:  Determine if this method is to be used and set pointer appropriately.
Code lines:  24
Contained by:  module  dynamical_friction_acceleration_chandrasekhar
Modules used:  input_parameters  iso_varying_string

file:  satellites.dynamical_friction.acceleration.F90
Description:  Contains a module that implements calculations of the acceleration due to dynamical friction for satellites.
Code lines:  113

module:  satellite_dynamical_friction
Description:  Implements calculations of dynamical friction for satellites.
Code lines:  91
Contained by:  file  satellites.dynamical_friction.acceleration.F90
Modules used:  iso_varying_string
19. Source Code Documentation

**Used by:** subroutine *node_component_satellite_orbiting_rate_compute*

**function:** *satellite_dynamical_friction_acceleration*

**Description:** Return the satellite acceleration due to dynamical friction for *thisNode* (in units of km/s/Gyr).

**Code lines:** 13

**Contained by:** module *satellite_dynamical_friction*

**Modules used:** *galacticus_nodes*

**subroutine:** *satellite_dynamical_friction_acceleration_initialize*

**Description:** Initialize the satellite dynamical friction acceleration module.

**Code lines:** 55

**Contained by:** module *satellite_dynamical_friction*

**Modules used:** *dynamical_friction_acceleration_chandrasekhar*

*dynamical_friction_acceleration_null*

*galacticus_error*

*input_parameters*

**file:** *satellites.dynamical_friction.acceleration.null.F90*

**Description:** Contains a module with a null implementation of calculations of satellite acceleration due to dynamical friction.

**Code lines:** 55

**module:** *dynamical_friction_acceleration_null*

**Description:** Implements null value of calculations of satellite acceleration due to dynamical friction.

**Code lines:** 33

**Contained by:** file *satellites.dynamical_friction.acceleration.null.F90*

**Used by:** subroutine *satellite_dynamical_friction_acceleration_initialize*

**function:** *satellite_dynamical_friction_acceleration_null*

**Description:** Return a null acceleration for satellites due to dynamical friction.

**Code lines:** 9

**Contained by:** module *dynamical_friction_acceleration_null*

**Modules used:** *galacticus_nodes*

**subroutine:** *satellite_dynamical_friction_acceleration_null_initialize*

**Description:** Determine if this method is to be used and set pointer appropriately.

**Code lines:** 9

**Contained by:** module *dynamical_friction_acceleration_null*

**Modules used:** *iso_varying_string*

**file:** *satellites.merging.mass_movements.Baugh2005.F90*

**Description:** Contains a module which implements the Baugh et al. [2005] model of mass movements during satellite mergers.

**Code lines:** 155

**module:** *satellite_merging_mass_movements_baugh2005*

**Description:** Implements the Baugh et al. [2005] model of mass movements during satellite mergers.
19.1. Program units

Code lines: 135
Modules used: satellite_merging_mass_movements_descriptors
Used by: subroutine satellite_merging_mass_movement

subroutine: satellite_merging_mass_movement_baugh2005
Description: Determine how different mass components should be redistributed as the result of a merger according to the model of Baugh et al. [2005].
Code lines: 47
Contained by: module satellite_merging_mass_movements_baugh2005
Modules used: galactic_structure_enclosed_masses galactic_structure_options galacticus_nodes

subroutine: satellite_merging_mass_movements_baugh2005_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 66
Contained by: module satellite_merging_mass_movements_baugh2005
Modules used: galacticus_error input_parameters iso_varying_string

file: satellites.merging.mass_movements.F90
Description: Contains a module which determines how mass is moved around as a consequence of a satellite merging event.
Code lines: 118

module: satellite_merging_mass_movements
Description: Determines how mass is moved around as a consequence of a satellite merging event.
Code lines: 98
Contained by: file satellites.merging.mass_movements.F90
Modules used: iso_varying_string
Used by: subroutine satellite_merger_process

subroutine: satellite_merging_mass_movement
Description: Returns descriptors of how gas and stars move as the result of a satellite merger.
Code lines: 64
Contained by: module satellite_merging_mass_movements
Modules used: galacticus_error galacticus_nodes input_parameters satellite_merging_mass_movements_baugh2005 satellite_merging_mass_movements_simple satellite_merging_mass_movements_very_simple

subroutine: satellite_merging_mass_movement_store
Description: Compute and store the mass movement descriptors for this satellite merger.
Code lines: 9
Contained by: module satellite_merging_mass_movements
Modules used: galacticus_nodes satellite_merging_mass_movements_descriptors
file: satellites.merging.mass.movements.descriptors.F90
Description: Contains a module which defines descriptors for satellite merger mass movements.
Code lines: 34

module: satellite.merging.mass.movements.descriptors
Description: Defines descriptors for satellite merger mass movements.
Code lines: 14
Contained by: file satellites.merging.mass.movements.descriptors.F90
Used by: subroutine node_component_age_statistics_standard_satellite_merger
        subroutine node_component_age_simple_satellite_merging
        subroutine node_component_disk_statistics_standard_satellite_merger
        subroutine node_component_disk_statistics_standard_satellite_merger
        module satellite.merging.mass.movements.movement_store
        subroutine satellite.merging_mass_movement
        module satellite.merging.mass.movements.very_simple
        subroutine satellite.merging_mass_movement
        subroutine satellite.merging_mass_movement_simple

file: satellites.merging.mass.movements.simple.F90
Description: Contains a module which implements a simple model of mass movements during satellite mergers.
Code lines: 120

module: satellite.merging.mass.movements.simple
Description: Implements a simple model of mass movements during satellite mergers.
Code lines: 100
Contained by: file satellites.merging.mass.movements.simple.F90
Modules used: satellite.merging.mass.movements.descriptors
Used by: subroutine satellite.merging_mass_movement

subroutine: satellite.merging.mass_movement
Description: Determine where stars and gas move as the result of a merger event using a simple algorithm.
Code lines: 34
Contained by: module satellite.merging.mass.movements.simple
Modules used: galactic_structure_enclosed_masses galactic_structure_options galacticus_nodes

subroutine: satellite.merging.mass_movement_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 44
19.1. Program units

**Contained by:** module `satellite_merging_mass_movements_simple`

**Modules used:**
- `galacticus_error`
- `input_parameters`
- `iso_varying_string`

**file:** `satellites.merging.mass_movements.very_simple.F90`

**Description:** Contains a module which implements a very simple model of mass movements during satellite mergers.

**Code lines:** 60

**module:** `satellite_merging_mass_movements_very_simple`

**Description:** Implements a very simple model of mass movements during satellite mergers.

**Code lines:** 40

**Contained by:** file `satellites.merging.mass_movements.very_simple.F90`

**Modules used:**
- `satellite_merging_mass_movements_descriptors`

**Used by:** subroutine `satellite_merging_mass_movement`

**subroutine:** `satellite_merging_mass_movement_very_simple`

**Description:** Determine where stars and gas move as the result of a merger event using a simple algorithm.

**Code lines:** 15

**Contained by:** module `satellite_merging_mass_movements_very_simple`

**Modules used:**
- `galacticus_nodes`

**subroutine:** `satellite_merging_mass_movements_very_simple_initialize`

**Description:** Test if this method is to be used and set procedure pointer appropriately.

**Code lines:** 9

**Contained by:** module `satellite_merging_mass_movements_very_simple`

**Modules used:**
- `iso_varying_string`

**file:** `satellites.merging.remnant_sizes.Cole2000.F90`

**Description:** Contains a module which implements the Cole et al. [2000] algorithm for merger remnant sizes.

**Code lines:** 167

**module:** `satellite_merging_remnant_sizes_cole2000`

**Description:** Implements the Cole et al. [2000] algorithm for merger remnant sizes.

**Code lines:** 147

**Contained by:** file `satellites.merging.remnant_sizes.Cole2000.F90`

**Used by:** subroutine `satellite_merging_remnant_size`

**subroutine:** `satellite_merging_remnant_size_cole2000`

**Description:** Compute the size of the merger remnant for `satelliteNode` using the Cole et al. [2000] algorithm.

**Code lines:** 106

**Contained by:** module `satellite_merging_remnant_sizes_cole2000`

**Modules used:**
- `galactic_structure_enclosed_masses`
- `galactic_structure_options`
- `galacticus_display`
- `galacticus_error`
- `galacticus_nodes`
- `iso_varying_string`
19. Source Code Documentation

```
numerical_comparison  numerical_constants_physical
satellite_merging_remnant_sizes_-
progenitors  satellite_merging_remnant_sizes_
properties  string_handling

subroutine: satellite_merging_remnant_sizes_cole2000_initialize
  Description: Test if this method is to be used and set procedure pointer appropriately.
  Code lines: 23
  Contained by: module satellite_merging_remnant_sizes_cole2000
  Modules used: input_parameters  iso_varying_string

  Description: Contains a module which implements the Covington et al. [2008] algorithm for merger remnant sizes.
  Code lines: 219

module: satellite_merging_remnant_sizes_covington2008
  Description: Implements the Covington et al. [2008] algorithm for merger remnant sizes.
  Code lines: 199
  Used by: subroutine satellite_merging_remnant_size_covington2008

subroutine: satellite_merging_remnant_size_covington2008
  Description: Compute the size of the merger remnant for thisNode using the Covington et al. [2008] algorithm.
  Code lines: 141
  Contained by: module satellite_merging_remnant_sizes_covington2008
  Modules used: dark_matter_halo_scales  galacticus_display
               galacticus_error  galacticus_nodes
               iso_varying_string  numerical_comparison
               numerical_constants_physical  satellite_merging_remnant_sizes_
               progenitors  string_handling

subroutine: satellite_merging_remnant_sizes_covington2008_initialize
  Description: Test if this method is to be used and set procedure pointer appropriately.
  Code lines: 34
  Contained by: module satellite_merging_remnant_sizes_covington2008
  Modules used: input_parameters  iso_varying_string

file: satellites.merging.remnant_sizes.F90
  Description: Contains a module which implements calculations of merger remnant sizes.
  Code lines: 106

module: satellite_merging_remnant_sizes
  Description: Implements calculations of merger remnant sizes.
  Code lines: 86
```
19.1. Program units

file satellites.merging.remnant_sizes.F90

Modules used: iso_varying_string

Use by: subroutine satellite_merger_process

subroutine satellite_merging_remnant_size
Description: Computes the size of a merger remnant.
Code lines: 62
Contained by: module satellite_merging_remnant_sizes
Modules used:
galacticus_error
galacticus_nodes
input_parameters
satellite_merging_remnant_sizes_-cole2000
satellite_merging_remnant_sizes_-covington2008

file: satellites.merging.remnant_sizes.null.F90

Description: Contains a module which implements a null algorithm for merger remnant sizes.
Code lines: 52

module satellite_merging_remnant_sizes_null
Description: Implements a null algorithm for merger remnant sizes.
Code lines: 32
Contained by: file satellites.merging.remnant_sizes.null.F90
Used by: subroutine satellite_merging_remnant_size

subroutine satellite_merging_remnant_size_null
Description: A null implementation of merger remnant size. Does nothing.
Code lines: 8
Contained by: module satellite_merging_remnant_sizes_null
Modules used:
galacticus_nodes

subroutine satellite_merging_remnant_sizes_null_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 9
Contained by: module satellite_merging_remnant_sizes_null
Modules used: iso_varying_string


Description: Contains a module which implements calculations of progenitor properties for merger remnant calculations using the algorithm of Cole et al. [2000].
Code lines: 260

module satellite_merging_remnant_progenitors_properties_cole2000
Description: Implements calculations of progenitor properties for merger remnant calculations using the algorithm of Cole et al. [2000].
Code lines: 239
Modules used:
galacticus_nodes
root_finder
Used by: subroutine satellite_merging_remnant_size
progenitor_properties
function: half_mass_radius_root_cole2000
Description: Function used in root finding for progenitor galaxy half-mass radii.
Code lines: 27
Contained by: module satellite_merging_remnant_progenitors_properties_cole2000
Modules used:
  galactic_structure_enclosed_masses   galactic_structure_options
  satellite_merging_mass_movements_-descriptors

subroutine: satellite_merging_remnant_progenitor_properties_cole2000
Description: Computes various properties of the progenitor galaxies useful for calculations of merger remnant sizes.
Code lines: 171
Contained by: module satellite_merging_remnant_progenitors_properties_cole2000
Modules used:
  galactic_structure_enclosed_masses   galactic_structure_options
  galactic_structure_radii   galacticus_error
  numerical_constants_physical   satellite_merging_mass_movements_-descriptors

subroutine: satellite_merging_remnant_progenitor_properties_cole2000_init
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 16
Contained by: module satellite_merging_remnant_progenitors_properties_cole2000
Modules used:
  array_utilities   galacticus_error
  iso_varying_string

file: satellites.merging.remnant_sizes.progenitor_properties.F90
Description: Contains a module which implements calculations for progenitor properties for merger remnant calculations.
Code lines: 100

module: satellite_merging_remnant_sizes_progenitors
Description: Implements calculations for progenitor properties for merger remnant calculations.
Code lines: 80
Contained by: file satellites.merging.remnant_sizes.progenitor_properties.F90
Modules used:
  iso_varying_string
Used by:
  subroutine satellite_merging_remnant_-size_cole2000
  subroutine satellite_merging_remnant_-size_covington2008

subroutine: satellite_merging_remnant_progenitor_properties
Description: Calculates progenitor properties for merger remnant calculations.
Code lines: 60
Contained by: module satellite_merging_remnant_sizes_progenitors
Modules used:
  galacticus_error   galacticus_nodes
  input_parameters   satellite_merging_remnant_-progenitors_properties_cole2000
  satellite_merging_remnant_-progenitors_properties_standard
file: satellites.merging.remnant_sizes.progenitor_properties.standard.F90
Description: Contains a module which implements standard calculations of progenitor properties for merger remnant calculations.
Code lines: 213

module: satellite_merging_remnant_progenitors_properties_standard
Description: Implements standard calculations of progenitor properties for merger remnant calculations.
Code lines: 193
Contained by: file satellites.merging.remnant_sizes.progenitor_properties.standard.F90
Used by: subroutine satellite_merging_remnant_progenitor_properties

subroutine: satellite_merging_remnant_progenitor_properties_standard
Description: Computes various properties of the progenitor galaxies useful for calculations of merger remnant sizes.
Code lines: 161
Contained by: module satellite_merging_remnant_progenitors_properties_standard
Modules used: galactic_structure_enclosed_masses galactic_structure_options
galactic_structure_radii galacticus_error
galacticus_nodes numerical_constants_physical
satellite_merging_mass_movements_descriptors

subroutine: satellite_merging_remnant_progenitor_properties_standard_init
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 17
Contained by: module satellite_merging_remnant_progenitors_properties_standard
Modules used: array_utilities galacticus_error
galacticus_nodes iso_varying_string

file: satellites.merging.remnant_sizes.properties.F90
Description: Contains a module which stores properties of merger remnants related to their size.
Code lines: 30

module: satellite_merging_remnant_sizes_properties
Description: Stores properties of merger remnants related to their size.
Code lines: 10
Contained by: file satellites.merging.remnant_sizes.properties.F90
Used by: subroutine node_component_spheroid_standard_satellite_merging subroutine satellite_merging_remnant_size_cole2000
subroutine satellite_merging_remnant_size_covington2008

Description: Implements calculations of satellite merging times using the Boylan-Kolchin et al. [2008] method.
Code lines: 120

function: boylankolchin2008defaultconstructor
### 19. Source Code Documentation

**Description:** Default constructor for the Boylan-Kolchin et al. [2008] merging timescale class.

**Code lines:** 8

**Contained by:** file `satellites.merging.timescale.Boylan-Kolchin2008.F90`

**Modules used:**
- `galacticus_display`
- `input_parameters`

**subroutine: boylankolchin2008destructor**

**Description:** Default constructor for the Boylan-Kolchin et al. [2008] merging timescale class.

**Code lines:** 7

**Contained by:** file `satellites.merging.timescale.Boylan-Kolchin2008.F90`

**function: boylankolchin2008timeuntilmerging**

**Description:** Return the timescale for merging satellites using the Boylan-Kolchin et al. [2008] method.

**Code lines:** 62

**Contained by:** file `satellites.merging.timescale.Boylan-Kolchin2008.F90`

**Modules used:**
- `dark_matter_halo_scales`
- `dark_matter_profiles`
- `dynamical_friction_timescale utility`
- `galacticus_error`
- `galacticus_nodes`
- `kepler_orbits`
- `satellite_orbits`

**interface: satellitemergingtimescalesboylankolchin2008**

**Description:** Constructors for the Boylan-Kolchin et al. [2008] merging timescale class.

**Code lines:** 3

**Contained by:** file `satellites.merging.timescale.Boylan-Kolchin2008.F90`

**file: satellites.merging.timescale.F90**

**Description:** Contains a module that provides and object that implements satellite merging timescales.

**Code lines:** 1453

**module: satellite_merging_timescales**

Code lines: 1433
Contained by: file satellites.merging.timescale.F90
Modules used: fgs
iso_varying_string
kepler_orbits
Used by: subroutine galacticus_state_retrieve
galacticus_nodes
subroutine galacticus_state_store
subroutine time_until_merging_subresolution
subroutine node_component_satellite_orbiting_create
subroutine node_component_satellite_standard_create
subroutine node_component_satellite_standard_virial_orbit_set
subroutine node_component_satellite_very_simple_create

function: boylankolchin2008defaultconstructor
19. Source Code Documentation

**Description:** Default constructor for the Boylan-Kolchin et al. [2008] merging timescale class.

**Code lines:** 8

**Contained by:** module `satellite_merging_timescales`

**Modules used:**
- `galacticus_display`
- `input_parameters`

**subroutine:** `boylankolchin2008destructor`

**Description:** Default constructor for the Boylan-Kolchin et al. [2008] merging timescale class.

**Code lines:** 7

**Contained by:** module `satellite_merging_timescales`

**function:** `boylankolchin2008timeuntilmerging`

**Description:** Return the timescale for merging satellites using the Boylan-Kolchin et al. [2008] method.

**Code lines:** 62

**Contained by:** module `satellite_merging_timescales`

**Modules used:**
- `dark_matter_halo_scales`
- `dark_matter_profiles`
- `dynamical_friction_timescale`
- `galacticus_error`
- `utilities`
- `galacticus_nodes`
- `kepler_orbits`
- `satellite_orbits`

**function:** `infinitedefaultconstructor`

**Description:** Default constructor for the infinite merging timescale class.

**Code lines:** 8

**Contained by:** module `satellite_merging_timescales`

**Modules used:**
- `galacticus_display`
- `input_parameters`

**subroutine:** `infinitedestructor`

**Description:** Default constructor for the infinite merging timescale class.

**Code lines:** 7

**Contained by:** module `satellite_merging_timescales`

**function:** `infinitemergetimeuntilmerging`

**Description:** Return a zero timescale for satellite merging.

**Code lines:** 12

**Contained by:** module `satellite_merging_timescales`

**Modules used:**
- `galacticus_nodes`
- `kepler_orbits`

**function:** `jiang2008constructor`

**Description:** Constructor for the Jiang et al. [2008] merging timescale class.

**Code lines:** 9

**Contained by:** module `satellite_merging_timescales`

**function:** `jiang2008defaultconstructor`

**Description:** Default constructor for the Jiang et al. [2008] merging timescale class.

**Code lines:** 34

**Contained by:** module `satellite_merging_timescales`

**Modules used:**
- `galacticus_display`
- `galacticus_error`
- `galacticus_nodes`
- `input_parameters`
19.1. Program units

subroutine: jiang2008destructor
Description: Destructor for the Jiang et al. [2008] merging timescale class.
Code lines: 10
Contained by: module satellite_merging_timescales
Modules used: gaussian_random

subroutine: jiang2008staterestore
Description: Write the stored snapshot of the random number state to file.
Code lines: 11
Contained by: module satellite_merging_timescales
Modules used: pseudo_random

subroutine: jiang2008statesnapshot
Description: Store a snapshot of the random number generator internal state.
Code lines: 8
Contained by: module satellite_merging_timescales

subroutine: jiang2008statestore
Description: Write the stored snapshot of the random number state to file.
Code lines: 11
Contained by: module satellite_merging_timescales
Modules used: pseudo_random

function: jiang2008timeuntilmerging
Description: Return the timescale for merging satellites using the Jiang et al. [2008] method.
Code lines: 66
Contained by: module satellite_merging_timescales
Modules used: dark_matter_halo_scales dark_matter_profiles
dynamical_friction_timescale_-
galacticus_error
utilities
gaussian_random satellite_orbits

function: laceycole1993defaultconstructor
Description: Default constructor for the Lacey and Cole [1993] merging timescale class.
Code lines: 8
Contained by: module satellite_merging_timescales
Modules used: galacticus_display input_parameters

subroutine: laceycole1993destructor
Description: Default constructor for the Lacey and Cole [1993] merging timescale class.
Code lines: 7
Contained by: module satellite_merging_timescales

function: laceycole1993timeuntilmerging
Description: Return the timescale for merging satellites using the Lacey and Cole [1993] method.
Code lines: 26
Contained by: module satellite_merging_timescales
### 19. Source Code Documentation

**Modules used:**
- dark_matter_halo_scales
- galacticus_nodes
- kepler_orbits

**function:** laceycole1993timeuntilmergingmassdependence

**Description:** Return the mass-dependent part of the timescale for merging satellites using the Lacey and Cole [1993] method.

**Code lines:** 30

**Contained by:** module satellite_merging_timescales

**Modules used:**
- dark_matter_halo_scales
- dynamical_friction_timescale_utilities
- galacticus_nodes

**function:** laceycole1993tormendefaultconstructor

**Description:** Default constructor for the Cole et al. [2000] merging timescale class.

**Code lines:** 7

**Contained by:** module satellite_merging_timescales

**subroutine:** laceycole1993tormendestructor

**Description:** Destructor for the Cole et al. [2000] merging timescale class.

**Code lines:** 10

**Contained by:** module satellite_merging_timescales

**Modules used:**
- gaussian_random

**subroutine:** laceycole1993tormenstaterestore

**Description:** Write the stored snapshot of the random number state to file.

**Code lines:** 11

**Contained by:** module satellite_merging_timescales

**Modules used:**
- pseudo_random

**subroutine:** laceycole1993tormenstatesnapshot

**Description:** Store a snapshot of the random number generator internal state.

**Code lines:** 8

**Contained by:** module satellite_merging_timescales

**subroutine:** laceycole1993tormenstatestore

**Description:** Write the stored snapshot of the random number state to file.

**Code lines:** 11

**Contained by:** module satellite_merging_timescales

**Modules used:**
- pseudo_random

**function:** laceycole1993torrentimeuntilmerging

**Description:** Return the timescale for merging satellites using the Lacey and Cole [1993] method with a parameterization of orbital parameters designed to fit the results of Tormen [1997] as described by Cole et al. [2000].

**Code lines:** 24

**Contained by:** module satellite_merging_timescales

**Modules used:**
- galacticus_nodes
- gaussian_random
- kepler_orbits

**function:** nulldefaultconstructor
19.1. Program units

Description: Default constructor for the null merging timescale class.
Code lines: 8
Contained by: module satellite_merging_timescales
Modules used: galacticus_display input_parameters

subroutine: nulldestructor
Description: Default constructor for the null merging timescale class.
Code lines: 7
Contained by: module satellite_merging_timescales

function: nulltimeuntilmerging
Description: Return a zero timescale for satellite merging.
Code lines: 11
Contained by: module satellite_merging_timescales
Modules used: galacticus_nodes kepler_orbits

function: presetdefaultconstructor
Description: Default constructor for the preset merging timescale class.
Code lines: 8
Contained by: module satellite_merging_timescales
Modules used: galacticus_display input_parameters

subroutine: presetdestructor
Description: Default constructor for the preset merging timescale class.
Code lines: 7
Contained by: module satellite_merging_timescales

function: presettimeuntilmerging
Description: Return the timescale for merging satellites using the preset value.
Code lines: 14
Contained by: module satellite_merging_timescales
Modules used: galacticus_nodes kepler_orbits

subroutine: satellite_merging_timescale_state_retrieve
Description: Retrieve the state from file.
Code lines: 10
Contained by: module satellite_merging_timescales

subroutine: satellite_merging_timescale_state_snapshot
Description: Retrieve the state from file.
Code lines: 8
Contained by: module satellite_merging_timescales

subroutine: satellite_merging_timescale_state_store
Description: Store the state to file.
Code lines: 10
Contained by: module satellite_merging_timescales

interface: satellite_merging_timescales
19. Source Code Documentation

Code lines: 3
Contained by: module satellite_merging_timescales

interface: satellitemergingtimescalesboylankolchin2008
Description: Constructors for the Boylan-Kolchin et al. [2008] merging timescale class.
Code lines: 3
Contained by: module satellite_merging_timescales

type: satellitemergingtimescalesclass
Code lines: 42
Contained by: module satellite_merging_timescales

function: satellitemergingtimescalesconstructordefault
Description: Return a pointer to the default satelliteMergingTimescales object.
Code lines: 8
Contained by: module satellite_merging_timescales

function: satellitemergingtimescalesconstructornamed
Description: Return a pointer to a newly created satelliteMergingTimescales object of the specified type.
Code lines: 79
Contained by: module satellite_merging_timescales
Modules used: galacticus_error iso_varying_string

interface: satellitemergingtimescalesinfinite
Description: Constructors for the infinite merging timescale class.
Code lines: 3
Contained by: module satellite_merging_timescales

subroutine: satellitemergingtimescalesinitialize
Description: Initialize the default satelliteMergingTimescales object.
Code lines: 96
Contained by: module satellite_merging_timescales
Modules used: galacticus_error input_parameters iso_varying_string

function: satellitemergingtimescalesisfinalizable
Description: Return true if this object can be finalized.
Code lines: 7
Contained by: module satellite_merging_timescales

interface: satellitemergingtimescalesjiang2008
Description: Constructors for the Jiang et al. [2008] merging timescale class.
Code lines: 4
Contained by: module satellite_merging_timescales

interface: satellitemergingtimescaleslaceycole1993
Description: Constructors for the Lacey and Cole [1993] merging timescale class.
Code lines: 3
19.1. Program units

**Contained by:** module satellite_merging_timescales

**interface:** satellitemergingtimescaleslaceycole1993tormen
*Description:* Constructors for the Cole et al. [2000] merging timescale class.
*Code lines:* 3
*Contained by:* module satellite_merging_timescales

**interface:** satellitemergingtimescalesnull
*Description:* Constructors for the null merging timescale class.
*Code lines:* 3
*Contained by:* module satellite_merging_timescales

**interface:** satellitemergingtimescalespreset
*Description:* Constructors for the preset merging timescale class.
*Code lines:* 3
*Contained by:* module satellite_merging_timescales

**subroutine:** satellitemergingtimescalesstaterestore
*Description:* Restore the state of the object from file.
*Code lines:* 9
*Contained by:* module satellite_merging_timescales
*Modules used:* fgsl

**subroutine:** satellitemergingtimescalesstatesnapshot
*Description:* Stores a snapshot of the object state.
*Code lines:* 6
*Contained by:* module satellite_merging_timescales

**subroutine:** satellitemergingtimescalesstatestore
*Description:* Store the state of the object to file.
*Code lines:* 9
*Contained by:* module satellite_merging_timescales
*Modules used:* fgsl

**function:** satellitemergingtimescalestimeuntilmergingnull
*Description:* Return the time (in Gyr) until the satellite will merge with its host given the current orbit.
*Code lines:* 9
*Contained by:* module satellite_merging_timescales
*Modules used:* galacticus_error

**interface:** satellitemergingtimescalesvillalobos2013
*Description:* Constructors for the Villalobos et al. [2013] merging timescale class.
*Code lines:* 4
*Contained by:* module satellite_merging_timescales

**interface:** satellitemergingtimescaleswetzelwhite2010
*Description:* Constructors for the Wetzel and White [2010] merging timescale class.
*Code lines:* 3
*Contained by:* module satellite_merging_timescales
function: villalobos2013defaultconstructor
Description: Default constructor for the Villalobos et al. [2013] merging timescale class.
Code lines: 46
Contained by: module satellite_merging_timescales
Modules used: galacticus_display input_parameters

subroutine: villalobos2013destructor
Description: Default constructor for the Villalobos et al. [2013] merging timescale class.
Code lines: 8
Contained by: module satellite_merging_timescales

function: villalobos2013genericconstructor
Description: Generic constructor for the Villalobos et al. [2013] merging timescale class.
Code lines: 11
Contained by: module satellite_merging_timescales

function: villalobos2013timeuntilmerging
Description: Return the timescale for merging satellites using the Villalobos et al. [2013] method.
Code lines: 22
Contained by: module satellite_merging_timescales
Modules used: cosmology_functions dynamical_friction_timescale_utilities galacticus_nodes kepler_orbits

function: wetzelwhite2010defaultconstructor
Description: Default constructor for the Wetzel and White [2010] merging timescale class.
Code lines: 8
Contained by: module satellite_merging_timescales
Modules used: galacticus_display input_parameters

subroutine: wetzelwhite2010destructor
Description: Default constructor for the Wetzel and White [2010] merging timescale class.
Code lines: 7
Contained by: module satellite_merging_timescales

function: wetzelwhite2010timeuntilmerging
Description: Return the timescale for merging satellites using the Wetzel and White [2010] method.
Code lines: 27
Contained by: module satellite_merging_timescales
Modules used: cosmology_functions dynamical_friction_timescale_utilities galacticus_nodes kepler_orbits

Description: Implements calculations of satellite merging times using the Jiang et al. [2008] method.
Code lines: 217
Modules used: fgsl
function: jiang2008constructor
Description: Constructor for the Jiang et al. [2008] merging timescale class.
Code lines: 9

function: jiang2008defaultconstructor
Description: Default constructor for the Jiang et al. [2008] merging timescale class.
Code lines: 34
Modules used: galacticus_display galacticus_error galacticus_nodes input_parameters

subroutine: jiang2008destructor
Description: Destructor for the Jiang et al. [2008] merging timescale class.
Code lines: 10
Modules used: gaussian_random

subroutine: jiang2008staterestore
Description: Write the stored snapshot of the random number state to file.
Code lines: 11
Modules used: pseudo_random

subroutine: jiang2008statesnapshot
Description: Store a snapshot of the random number generator internal state.
Code lines: 8

subroutine: jiang2008statestore
Description: Write the stored snapshot of the random number state to file.
Code lines: 11
Modules used: pseudo_random

function: jiang2008timeuntilmerging
Description: Return the timescale for merging satellites using the Jiang et al. [2008] method.
Code lines: 66
Modules used: dark_matter_halo_scales dark_matter_profiles dynamical_friction_timescale_Utilities galacticus_error galacticus_nodes gaussian_random satellite_orbits

interface: satellitesmergingtimescalesjiang2008
Description: Constructors for the Jiang et al. [2008] merging timescale class.
Code lines: 4
19. Source Code Documentation

**file:** satellites.merging.timescale.Lacey-Cole.F90  
**Description:** Implements calculations of satellite merging times using the Lacey and Cole [1993] method.  
**Code lines:** 126

**function:** laceycole1993defaultconstructor  
**Description:** Default constructor for the Lacey and Cole [1993] merging timescale class.  
**Code lines:** 8  
**Contained by:** file satellites.merging.timescale.Lacey-Cole.F90  
**Modules used:** galacticus_display input_parameters

**subroutine:** laceycole1993destructor  
**Description:** Default constructor for the Lacey and Cole [1993] merging timescale class.  
**Code lines:** 7  
**Contained by:** file satellites.merging.timescale.Lacey-Cole.F90

**function:** laceycole1993timeuntilmerging  
**Description:** Return the timescale for merging satellites using the Lacey and Cole [1993] method.  
**Code lines:** 26  
**Contained by:** file satellites.merging.timescale.Lacey-Cole.F90  
**Modules used:** dark_matter_halo_scales galacticus_nodes kepler_orbits

**function:** laceycole1993timeuntilmergingmassdependence  
**Description:** Return the mass-dependent part of the timescale for merging satellites using the Lacey and Cole [1993] method.  
**Code lines:** 30  
**Contained by:** file satellites.merging.timescale.Lacey-Cole.F90  
**Modules used:** dark_matter_halo_scales dynamical_friction_timescale_-utilities galacticus_nodes

**interface:** satellitemergingtimescaleslaceycole1993  
**Description:** Constructors for the Lacey and Cole [1993] merging timescale class.  
**Code lines:** 3  
**Contained by:** file satellites.merging.timescale.Lacey-Cole.F90

**file:** satellites.merging.timescale.Lacey-Cole_Tormen.F90  
**Description:** Implements calculations of satellite merging times using the Lacey and Cole [1993] method with a parameterization of orbital parameters designed to fit the results of Tormen [1997] as described by Cole et al. [2000].  
**Code lines:** 128  
**Modules used:** fgsl

**function:** laceycole1993tormendefaultconstructor  
**Description:** Default constructor for the Cole et al. [2000] merging timescale class.  
**Code lines:** 7  
**Contained by:** file satellites.merging.timescale.Lacey-Cole_Tormen.F90
subroutine: laceycole1993tormendestructor
Description: Destructor for the Cole et al. [2000] merging timescale class.
Code lines: 10
Contained by: file satellites.merging.timescale.Lacey-Cole_Tormen.F90
Modules used: gaussian_random

subroutine: laceycole1993tormenstaterestore
Description: Write the stored snapshot of the random number state to file.
Code lines: 11
Contained by: file satellites.merging.timescale.Lacey-Cole_Tormen.F90
Modules used: pseudo_random

subroutine: laceycole1993tormenstatesnapshot
Description: Store a snapshot of the random number generator internal state.
Code lines: 8
Contained by: file satellites.merging.timescale.Lacey-Cole_Tormen.F90

subroutine: laceycole1993tormenstatestore
Description: Write the stored snapshot of the random number state to file.
Code lines: 11
Contained by: file satellites.merging.timescale.Lacey-Cole_Tormen.F90
Modules used: pseudo_random

function: laceycole1993tormentimeuntilmerging
Description: Return the timescale for merging satellites using the Lacey and Cole [1993] method with a parameterization of orbital parameters designed to fit the results of Tormen [1997] as described by Cole et al. [2000].
Code lines: 24
Contained by: file satellites.merging.timescale.Lacey-Cole_Tormen.F90
Modules used: galacticus_nodes gaussian_random kepler_orbits

interface: satellitemergingtimescaleslaceycole1993tormen
Description: Constructors for the Cole et al. [2000] merging timescale class.
Code lines: 3
Contained by: file satellites.merging.timescale.Lacey-Cole_Tormen.F90

file: satellites.merging.timescale.Villalobos_2013.F90
Description: Implements calculations of satellite merging times by applying the Villalobos et al. [2013] modifier to another selected satellite merging time method.
Code lines: 143

interface: satellitemergingtimescalesvillalobos2013
Description: Constructors for the Villalobos et al. [2013] merging timescale class.
Code lines: 4
Contained by: file satellites.merging.timescale.Villalobos_2013.F90
function: villalobos2013defaultconstructor
Description: Default constructor for the Villalobos et al. [2013] merging timescale class.
Code lines: 46
Contained by: file satellites.merging.timescale.Villalobos_2013.F90
Modules used: galacticus_display input_parameters

subroutine: villalobos2013destructor
Description: Default constructor for the Villalobos et al. [2013] merging timescale class.
Code lines: 8
Contained by: file satellites.merging.timescale.Villalobos_2013.F90

function: villalobos2013genericconstructor
Description: Generic constructor for the Villalobos et al. [2013] merging timescale class.
Code lines: 11
Contained by: file satellites.merging.timescale.Villalobos_2013.F90

function: villalobos2013timeuntilmerging
Description: Return the timescale for merging satellites using the Villalobos et al. [2013] method.
Code lines: 22
Contained by: file satellites.merging.timescale.Villalobos_2013.F90
Modules used: cosmology_functions dynamical_friction_timescale_utilities galacticus_nodes kepler_orbits

file: satellites.merging.timescale.Wetzel-White.F90
Description: Implements calculations of satellite merging times using the Wetzel and White [2010] method.
Code lines: 87

interface: satellitemergingtimescaleswetzelwhite2010
Description: Constructors for the Wetzel and White [2010] merging timescale class.
Code lines: 3
Contained by: file satellites.merging.timescale.Wetzel-White.F90

function: wetzelwhite2010defaultconstructor
Description: Default constructor for the Wetzel and White [2010] merging timescale class.
Code lines: 8
Contained by: file satellites.merging.timescale.Wetzel-White.F90
Modules used: galacticus_display input_parameters

subroutine: wetzelwhite2010destructor
Description: Default constructor for the Wetzel and White [2010] merging timescale class.
Code lines: 7
Contained by: file satellites.merging.timescale.Wetzel-White.F90

function: wetzelwhite2010timeuntilmerging
Description: Return the timescale for merging satellites using the Wetzel and White [2010] method.
Code lines: 27
Contained by: file satellites.merging.timescale.Wetzel-White.F90
19.1. Program units

**Modules used:**
- cosmology_functions
- dynamical_friction_timescale_utilities
- galacticus_nodes
- kepler_orbits

**file:** satellites.merging.timescale.infinite.F90
**Description:** Implements calculations of satellite merging times that are always infinite.
**Code lines:** 73

**function:** infinitedefaultconstructor
**Description:** Default constructor for the infinite merging timescale class.
**Code lines:** 8
**Contained by:** file satellites.merging.timescale.infinite.F90
**Modules used:** galacticus_display input_parameters

**subroutine:** infinitedestructor
**Description:** Default constructor for the infinite merging timescale class.
**Code lines:** 7
**Contained by:** file satellites.merging.timescale.infinite.F90

**function:** infinitetimeuntilmerging
**Description:** Return a zero timescale for satellite merging.
**Code lines:** 12
**Contained by:** file satellites.merging.timescale.infinite.F90
**Modules used:** galacticus_nodes kepler_orbits

**interface:** satellitemergingtimescalesinfinite
**Description:** Constructors for the infinite merging timescale class.
**Code lines:** 3
**Contained by:** file satellites.merging.timescale.infinite.F90

**file:** satellites.merging.timescale.null.F90
**Description:** Implements calculations of satellite merging times that are always zero.
**Code lines:** 69

**function:** nulldefaultconstructor
**Description:** Default constructor for the null merging timescale class.
**Code lines:** 8
**Contained by:** file satellites.merging.timescale.null.F90
**Modules used:** galacticus_display input_parameters

**subroutine:** nulldestructor
**Description:** Default constructor for the null merging timescale class.
**Code lines:** 7
**Contained by:** file satellites.merging.timescale.null.F90

**function:** nulltimeuntilmerging
**Description:** Return a zero timescale for satellite merging.
**Code lines:** 11
**Contained by:** file satellites.merging.timescale.null.F90
19. **Source Code Documentation**

**Modules used:** galacticus_nodes kepler_orbits

**interface:** satellitesmergingtimescalesnull

*Description:* Constructors for the null merging timescale class.

*Code lines:* 3

*Contained by:* file satellites.merging.timescale.null.F90

**file:** satellites.merging.timescale.preset.F90

*Description:* Implements calculations of satellite merging times using preset values.

*Code lines:* 72

**function:** presetdefaultconstructor

*Description:* Default constructor for the preset merging timescale class.

*Code lines:* 8

*Contained by:* file satellites.merging.timescale.preset.F90

*Modules used:* galacticus_display input_parameters

**subroutine:** presetdestructor

*Description:* Default constructor for the preset merging timescale class.

*Code lines:* 7

*Contained by:* file satellites.merging.timescale.preset.F90

**function:** presettimeuntilmerging

*Description:* Return the timescale for merging satellites using the preset value.

*Code lines:* 14

*Contained by:* file satellites.merging.timescale.preset.F90

*Modules used:* galacticus_nodes kepler_orbits

**interface:** satellitesmergingtimescalespreset

*Description:* Constructors for the preset merging timescale class.

*Code lines:* 3

*Contained by:* file satellites.merging.timescale.preset.F90

**file:** satellites.merging.timescale.utilities.F90

*Description:* Contains a module which implements utilities for dynamical friction timescale calculations.

*Code lines:* 64

**module:** dynamical_friction_timescale_utilities

*Description:* Implements utilities for dynamical friction timescale calculations.

*Code lines:* 44

*Contained by:* file satellites.merging.timescale.utilities.F90

*Used by:*

- function boylankolchin2008timeuntilmerging
- function jiang2008timeuntilmerging
- function villalobos2013timeuntilmerging
- function wetzelwhite2010timeuntilmerging
function **jiang2008timeuntilmerging** function**laceycole1993timeuntilmergingmassdependence** function**villalobos2013timeuntilmerging** function**wetzelwhite2010timeuntilmerging**

**function: dynamical_friction_timescale_multiplier**

*Description:* Returns a multiplicative factor for scaling of dynamical friction timescales for satellite merging time calculations.

*Code lines:* 28

*Contained by:* module **dynamical_friction_timescale_utilities**

*Modules used:* **input_parameters**


*Description:* Contains a module which implements the Benson [2005] orbital parameter distribution for merging subhalos.

*Code lines:* 144

**module: virial_orbits_benson2005**

*Description:* Implements the Benson [2005] orbital parameter distribution for merging subhalos.

*Code lines:* 124

*Contained by:* file **satellites.merging.virial_orbits.Benson2005.F90**

*Modules used:* **fgsl**

*Used by:* subroutine **galacticus_state_retrieve** subroutine **galacticus_state_snapshot** subroutine **galacticus_state_store** function **virial_orbital_parameters**

**function: virial_orbital_parameters_benson2005**

*Description:* Return orbital parameters of a satellite selected at random from the fitting function found by Benson [2005].

*Code lines:* 54

*Contained by:* module **virial_orbits_benson2005**

*Modules used:* **dark_matter_halo_scales** **galacticus_error** **galacticus_nodes** **kepler_orbits** **pseudo_random**

**subroutine: virial_orbital_parameters_benson2005_initialize**

*Description:* Test if this method is to be used and set procedure pointer appropriately.

*Code lines:* 9

*Contained by:* module **virial_orbits_benson2005**

*Modules used:* **iso_varying_string**

**subroutine: virial_orbital_parameters_benson2005_snapshot**

*Description:* Store a snapshot of the random number generator internal state.

*Code lines:* 7

*Contained by:* module **virial_orbits_benson2005**

**subroutine: virial_orbital_parameters_benson2005_state_retrieve**

*Description:* Write the stored snapshot of the random number state to file.

*Code lines:* 10

*Contained by:* module **virial_orbits_benson2005**
19. Source Code Documentation

Modules used: pseudo_random

subroutine: virial_orbital_parameters_benson2005_state_store
Description: Write the stored snapshot of the random number state to file.
Code lines: 10
Contained by: module virial_orbits_benson2005
Modules used: pseudo_random

file: satellites.merging.virial_orbits.F90
Description: Contains a module which implements satellite orbital parameters at virial radius crossing.
Code lines: 105

module: virial_orbits
Description: Implements satellite orbital parameters at virial radius crossing.
Code lines: 85
Contained by: file satellites.merging.virial_orbits.F90
Modules used: galacticus_nodes iso_varying_string
Used by: subroutine scan_for_mergers subroutine node_component_satellite_orbiting_create
subroutine node_component_satellite_standard_create function node_component_satellite_standard_virial_orbit
subroutine node_component_satellite_virial_orbits
function: virial_orbital_parameters
Description: Returns virial orbital parameters.
Code lines: 64
Contained by: module virial_orbits
Modules used: galacticus_error input_parameters
kepler_orbits virial_orbits_benson2005
virial_orbits_fixed virial_orbits_wetzel2010

Description: Contains a module which implements the Wetzel [2010] orbital parameter distribution for merging subhalos.
Code lines: 241

module: virial_orbits_wetzel2010
Description: Implements the Wetzel [2010] orbital parameter distribution for merging subhalos.
Code lines: 221
Contained by: file satellites.merging.virial_orbits.Wetzel2010.F90
Modules used: fgsl tables
Used by: subroutine galacticus_state_retrieve subroutine galacticus_state_snapshot
subroutine galacticus_state_store function virial_orbital_parameters

function: circularity_cumulative_probability
Description: The cumulative probability distribution for orbital circularity.
Code lines: 8
Contained by: module virial_orbits_wetzel2010
19.1. Program units

Modules used: hypergeometric_functions

function: circularity_root
Description: Function used in finding the circularity corresponding to a given cumulative probability.
Code lines: 8
Contained by: module virial_orbits_wetzel2010

function: virial_orbital_parameters_wetzel2010
Description: Return orbital velocities of a satellite selected at random from the fitting function found by Wetzel [2010].
Code lines: 92
Contained by: module virial_orbits_wetzel2010
Modules used: cosmology_functions critical_overdensity
dark_matter_halo_scales galacticus_nodes
kepler_orbits pseudo_random
root_finder

subroutine: virial_orbital_parameters_wetzel2010_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 31
Contained by: module virial_orbits_wetzel2010
Modules used: hypergeometric_functions iso_varying_string

subroutine: virial_orbital_parameters_wetzel2010_snapshot
Description: Store a snapshot of the random number generator internal state.
Code lines: 7
Contained by: module virial_orbits_wetzel2010

subroutine: virial_orbital_parameters_wetzel2010_state_retrieve
Description: Write the stored snapshot of the random number state to file.
Code lines: 10
Contained by: module virial_orbits_wetzel2010
Modules used: pseudo_random

subroutine: virial_orbital_parameters_wetzel2010_state_store
Description: Write the stored snapshot of the random number state to file.
Code lines: 10
Contained by: module virial_orbits_wetzel2010
Modules used: pseudo_random

file: satellites.merging.virial_orbits.fixed.F90
Description: Contains a module which implements a fixed orbital parameter distribution for merging subhalos.
Code lines: 99

module: virial_orbits_fixed
Description: Implements a fixed orbital parameter distribution for merging subhalos.
Code lines: 79
Contained by: file satellites.merging.virial_orbits.fixed.F90
19. Source Code Documentation

Used by: function `virial_orbital_parameters`

**function: virial_orbital_parameters_fixed**
*Description:* Return fixed orbital parameters for a satellite.
*Code lines:* 25
*Contained by:* module `virial_orbits_fixed`
*Modules used:* `dark_matter_halo_scales`, `galacticus_nodes`, `kepler_orbits`

**subroutine: virial_orbital_parameters_fixed_initialize**
*Description:* Test if this method is to be used and set procedure pointer appropriately.
*Code lines:* 36
*Contained by:* module `virial_orbits_fixed`
*Modules used:* `input_parameters`, `iso_varying_string`

**file: satellites.orbits.F90**
*Description:* Contains a module which implements calculations related to satellite orbits.
*Code lines:* 232

**module: satellite_orbits**
*Description:* Implements calculations related to satellite orbits.
*Code lines:* 212
*Contained by:* file `satellites.orbits.F90`
*Modules used:* `galacticus_nodes`
*Used by:* subroutine `galacticus_output_tree_satellite_extremum`, subroutine `galacticus_output_tree_satellite_host`, function `hot_halo_ram_pressure_force_font2008_get`, subroutine `node_component_dynamics_statistics_bars_record`, function `boylankolchin2008timeuntilmerging`, function `jiang2008timeuntilmerging`, function `satellites_tidal_fields_spherical_symmetry_get`

**function: equivalent_circular_orbit_solver**
*Description:* Root function used in finding equivalent circular orbits.
*Code lines:* 26
*Contained by:* module `satellite_orbits`
*Modules used:* `dark_matter_profiles`, `dark_matter_profiles_error_codes`, `galacticus_error`

**function: extremum_solver**
*Description:* Root function used in finding orbital extremum radius.
*Code lines:* 10
*Contained by:* module `satellite_orbits`
*Modules used:* `dark_matter_profiles`
19.1. Program units

Description: Takes a virial orbit and adjusts the energy to account for the change in the definition of potential between the original halo in which the orbit was defined and the current halo. Since the potential at the virial radius of halos is always defined to be $\Phi(r_{\text{vir}}) = -V_{\text{vir}}^2$, then the specific energy transforms as:

$$e \rightarrow e + V_{\text{vir},0}^2 + \Phi(r_{\text{vir},0}),$$

(19.29)

where subscript 0 refers to the original halo in which the orbit was defined and $\Phi(r)$ is the potential of the current halo.
19. Source Code Documentation

Code lines: 26
Contained by: module satellite_orbits
Modules used: galactic_structure_potentials, kepler_orbits

function: satellite_orbit_equivalent_circular_orbit_radius
Description: Solves for the equivalent circular orbit radius for thisOrbit in hostNode.
Code lines: 42
Contained by: module satellite_orbits
Modules used: dark_matter_halo_scales, kepler_orbits

subroutine: satellite_orbit_extremum_phase_space_coordinates
Description: Solves for the pericentric radius and velocity of thisOrbit in hostNode.
Code lines: 73
Contained by: module satellite_orbits
Modules used: dark_matter_profiles, dark_matter_profiles_error_codes, galacticus_error, kepler_orbits, numerical_constants_physical, numerical_constants_prefixes, root_finder

file: satellites.promotion.F90
Description: Contains a module which handles events where a satellite is moved to a new host halo.
Code lines: 89

module: satellite_promotion
Description: Handles events where a satellite is moved to a new host halo.
Code lines: 69
Contained by: file satellites.promotion.F90
Used by: subroutine events_node_merger_do_slh

subroutine: satellite_move_to_new_host
Description: Move satelliteNode to be a satellite of newHostNode.
Code lines: 59
Contained by: module satellite_promotion
Modules used: galacticus_display, galacticus_nodes, iso_varying_string, node_component_satellite_standard, node_component_satellite_very_simple, string_handling

file: satellites.tidal_fields.F90
Description: Contains a module that implements calculations of tidal fields acting on satellites.
Code lines: 100

module: satellites_tidal_fields
Description: Implements calculations of tidal fields acting on satellites.
Code lines: 80
Contained by: file satellites.tidal_fields.F90
Modules used: iso_varying_string
Used by: subroutine bar_instability_timescale_eln_tidal, subroutine node_component_spheroid_standard_rate_compute
19.1. Program units

function tidal_stripping_mass_loss_rate_disk_simple
function tidal_stripping_mass_loss_rate_spheroid_simple

function: satellite_tidal_field
Description: Return the tidal field acting on a satellite thisNode.
Code lines: 60
Contained by: module satellites_tidal_fields
Modules used: galacticus_error
input_parameters satellites_tidal_fields_null
satellites_tidal_fields_spherical_symmetry
ile: satellites.tidal_fields.null.F90
Description: Contains a module which implements a null tidal field for satellites.
Code lines: 54

module: satellites_tidal_fields_null
Description: Implements a null tidal field for satellites.
Code lines: 34
Contained by: file satellites.tidal_fields.null.F90
Modules used: galacticus_nodes
Used by: function satellite_tidal_field

function: satellites_tidal_fields_null_get
Description: Computes the tidal field acting on a satellite in the null implementation. Always returns zero.
Code lines: 8
Contained by: module satellites_tidal_fields_null
Modules used: galacticus_nodes

subroutine: satellites_tidal_fields_null_initialize
Description: Initializes the “Null” hot halo ram pressure stripping module.
Code lines: 10
Contained by: module satellites_tidal_fields_null
Modules used: input_parameters iso_varying_string

file: satellites.tidal_fields.spherical_symmetry.F90
Description: Contains a module which implements a model of the tidal field acting on a satellite assuming spherical symmetry in the host.
Code lines: 100

module: satellites_tidal_fields_spherical_symmetry
Description: Implements a module which implements a model of the tidal field acting on a satellite assuming spherical symmetry in the host.
Code lines: 80
Contained by: file satellites.tidal_fields.spherical_symmetry.F90
Modules used: galacticus_nodes
Used by: function satellite_tidal_field

function: satellites_tidal_fields_spherical_symmetry_get
19. Source Code Documentation

**Description:** Computes the tidal field acting on a satellite assuming a spherically symmetric host.

**Code lines:** 38

**Contained by:** module `satellites_tidal_fields_spherical_symmetry`

**Modules used:**
- `galactic_structure_densities`
- `galactic_structure_enclosed_masses`
- `galactic_structure_options`
- `galactic_nodes`
- `kepler_orbits`
- `numerical_constants_math`
- `numerical_constants_physical`
- `satellite_orbits`

**subroutine:** `satellites_tidal_fields_spherical_symmetry_initialize`

**Description:** Initializes the “spherical symmetry” satellite tidal field module.

**Code lines:** 23

**Contained by:** module `satellites_tidal_fields_spherical_symmetry`

**Modules used:**
- `input_parameters`
- `iso_varying_string`

**file:** `satellites.tidal_heating.rate.F90`

**Description:** Contains a module that implements calculations of the tidal heating rate for satellites.

**Code lines:** 113

**module:** `satellite_tidal_heating`

**Description:** Implements calculations of tidal heating for satellites.

**Code lines:** 91

**Contained by:** file `satellites.tidal_heating.rate.F90`

**Modules used:**
- `iso_varying_string`

**Used by:** subroutine `node_component_satellite_orbiting_rate_compute`

**function:** `satellite_tidal_heating_rate`

**Description:** Return the satellite tidal heating rate for `thisNode` (in units of (km/s/Mpc)^2/Gyr).

**Code lines:** 12

**Contained by:** module `satellite_tidal_heating`

**Modules used:**
- `galacticus_nodes`

**subroutine:** `satellite_tidal_heating_rate_initialize`

**Description:** Initialize the satellite tidal heating rate module.

**Code lines:** 56

**Contained by:** module `satellite_tidal_heating`

**Modules used:**
- `galacticus_error`
- `input_parameters`
- `tidal_heating_rate_gnedin`
- `tidal_heating_rate_null`

**file:** `satellites.tidal_heating.rate.Gnedin1999.F90`

**Description:** Contains a module with a Gnedin et al. [1999] implementation of calculations of satellite tidal heating.

**Code lines:** 122

**module:** `tidal_heating_rate_gnedin`

**Description:** Implements Gnedin et al. [1999] value of calculations of satellite tidal heating.

**Code lines:** 100

**Contained by:** file `satellites.tidal_heating.rate.Gnedin1999.F90`

**Used by:** subroutine `satellite_tidal_heating_rate_initialize`
function: satellite_tidal_heating_rate_gnedin
Description: Return the Gnedin et al. [1999] rate for satellite tidal heating.
Code lines: 46
Contained by: module tidal_heating_rate_gnedin
Modules used: dark_matter_halo_scales error_functions
galactic_structure_densities galactic_structure_enclosed_masses
galactic_structure_options galacticus_nodes
numerical_constants_astronomical numerical_constants_math
numerical_constants_physical numerical_constants_prefixes
tensors vectors

subroutine: satellite_tidal_heating_rate_gnedin_initialize
Description: Determine if this method is to be used and set pointer appropriately.
Code lines: 36
Contained by: module tidal_heating_rate_gnedin
Modules used: input_parameters iso_varying_string

file: satellites.tidal_heating.rate.null.F90
Description: Contains a module with a null implementation of calculations of satellite tidal heating.
Code lines: 54

module: tidal_heating_rate_null
Description: Implements null calculations of satellite tidal heating.
Code lines: 32
Contained by: file satellites.tidal_heating.rate.null.F90
Used by: subroutine satellite_tidal_heating_rate_initialize

function: satellite_tidal_heating_rate_null
Description: Return the null rate for satellite tidal heating.
Code lines: 8
Contained by: module tidal_heating_rate_null
Modules used: galacticus_nodes

subroutine: satellite_tidal_heating_rate_null_initialize
Description: Determine if this method is to be used and set pointer appropriately.
Code lines: 9
Contained by: module tidal_heating_rate_null
Modules used: iso_varying_string

file: satellites.tidal_stripping.rate.F90
Description: Contains a module that implements calculations of the mass loss rate due to tidal stripping for satellites.
Code lines: 113

module: satellite_tidal_stripping
Description: Implements calculations of tidal stripping for satellites.
Code lines: 90
19. Source Code Documentation

**Contained by:** file `satellites.tidal_stripping.rate.F90`

**Modules used:** `iso_varying_string`

**Used by:** subroutine `node_component_satellite_orbiting_rate_compute`

**subroutine: satellite_tidal_stripping_initialize**

**Description:** Initialize the satellite tidal stripping rate module.

**Code lines:** 56

**Contained by:** module `satellite_tidal_stripping`

**Modules used:**
- `galacticus_error`
- `input_parameters`
- `tidal_stripping_rate_null`
- `tidal_stripping_rate_zentner2005`

**function: satellite_tidal_stripping_rate**

**Description:** Return the satellite mass loss rate due to tidal stripping for thisNode (in units of $M_\odot$/Gyr).

**Code lines:** 12

**Contained by:** module `satellite_tidal_stripping`

**Modules used:** `galacticus_nodes`


**Description:** Contains a module with a King [1962] implementation of calculations of satellite mass loss due to tidal stripping.

**Code lines:** 147

**module: tidal_stripping_rate_zentner2005**

**Description:** Implements the King [1962] calculation of satellite mass loss due to tidal stripping.

**Code lines:** 124

**Contained by:** file `satellites.tidal_stripping.rate.Zentner2005.F90`

**Modules used:**
- `dark_matter_halo_scales`
- `error_functions`
- `galactic_structure_densities`
- `galactic_structure_enclosed_masses`
- `galactic_structure_options`
- `galacticus_nodes`
- `numerical_constants_astronomical`
- `numerical_constants_math`
- `numerical_constants_physical`
- `numerical_constants_prefixes`
- `root_finder`
- `vectors`

**function: satellite_tidal_stripping_rate_zentner2005**

**Description:** Return the Zentner2005 mass loss rate for satellites due to tidal stripping.

**Code lines:** 58

**Contained by:** module `tidal_stripping_rate_zentner2005`

**Modules used:**
- `input_parameters`

**subroutine: satellite_tidal_stripping_rate_zentner2005_initialize**

**Description:** Determine if this method is to be used and set pointer appropriately.

**Code lines:** 24

**Contained by:** module `tidal_stripping_rate_zentner2005`

**Modules used:**
- `input_parameters`
- `iso_varying_string`
function: tidal_radius_heated_halo_solver
Description: Root function used to find the tidal radius within a subhalo prior to heating.
Code lines: 16
Contained by: module tidal_stripping_rate_zentner2005
Modules used: galactic_structure_enclosed_masses numerical_constants_astronomical
numerical_constants_physical numerical_constants_prefixes

file: satellites.tidal_stripping.rate.null.F90
Description: Contains a module with a null implementation of calculations of satellite mass loss due to tidal stripping.
Code lines: 54

module: tidal_stripping_rate_null
Description: Implements null value of calculations of satellite mass loss due to tidal stripping.
Code lines: 32
Contained by: file satellites.tidal_stripping.rate.null.F90
Used by: subroutine satellite_tidal_stripping_rate_null Initialize

function: satellite_tidal_stripping_rate_null
Description: Return the null mass loss for satellites due to tidal stripping.
Code lines: 8
Contained by: module tidal_stripping_rate_null
Modules used: galacticus_nodes

subroutine: satellite_tidal_stripping_rate_null_initialize
Description: Determine if this method is to be used and set pointer appropriately.
Code lines: 9
Contained by: module tidal_stripping_rate_null
Modules used: iso_varying_string

file: shocks.one_dimensional.F90
Description: Contains a module which implements calculations of one-dimensional shocks.
Code lines: 43

module: shocks_1d
Description: Implements calculations of one-dimensional shocks.
Code lines: 23
Contained by: file shocks.one_dimensional.F90
Used by: function coldmodecoldmodefraction function coldmodecoldmodefraction

function: shocks_1d_density_jump
Description: Computes the density jump across a one-dimensional shock.
Code lines: 11
Contained by: module shocks_1d

file: star_formation.IMF.Baugh2005TopHeavy.F90
Description: Contains a module which implements the top-heavy stellar initial mass function from Baugh et al. [2005].
module: star_formation_imf_baugh2005topheavy
Description: Implements the top-heavy stellar initial mass function from Baugh et al. [2005].
Code lines: 211
Contained by: file star_formation.IMF.Baugh2005TopHeavy.F90
Used by: module star_formation_imf

subroutine: star_formation_imf_initialize_baugh2005topheavy
Description: Initialize the Baugh2005TopHeavy IMF module.
Code lines: 42
Contained by: module star_formation_imf_baugh2005topheavy

subroutine: star_formation_imf_maximum_mass_baugh2005topheavy
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_baugh2005topheavy

subroutine: star_formation_imf_minimum_mass_baugh2005topheavy
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_baugh2005topheavy

subroutine: star_formation_imf_phi_baugh2005topheavy
Description: Register the name of this IMF.
Code lines: 15
Contained by: module star_formation_imf_baugh2005topheavy
Modules used: star_formation_imf_ppl

subroutine: star_formation_imf_recycled_instantaneous_baugh2005topheavy
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_baugh2005topheavy

subroutine: star_formation_imf_register_baugh2005topheavy
Description: Register this IMF by incrementing the count and keeping a record of the assigned index.
Code lines: 8
Contained by: module star_formation_imf_baugh2005topheavy

subroutine: star_formation_imf_register_name_baugh2005topheavy
Description: Register the name of this IMF.
Code lines: 9
Contained by: module star_formation_imf_baugh2005topheavy
Modules used: iso_varying_string

subroutine: star_formation_imf_tabulate_baugh2005topheavy
Description: Register the name of this IMF.
Code lines: 21
19.1. Program units

*Contained by:* module `star_formation_imf_baugh2005topheavy`
*Modules used:* `star_formation_imf_ppl`  `tables`

**subroutine:** `star_formation_imf_yield_instantaneous_baugh2005topheavy`
*Description:* Register the name of this IMF.
*Code lines:* 13
*Contained by:* module `star_formation_imf_baugh2005topheavy`

**file:** `star_formation.IMF.Chabrier.F90`
*Description:* Contains a module which implements the Chabrier stellar initial mass function [Chabrier, 2001].
*Code lines:* 240

**module:** `star_formation_imf_chabrier`
*Description:* Implements the Chabrier stellar initial mass function.
*Code lines:* 220
*Contained by:* file `star_formation.IMF.Chabrier.F90`
*Used by:* module `star_formation_imf`

**function:** `chabrier_phi`
*Description:* Evaluates the Chabrier initial mass function.
*Code lines:* 13
*Contained by:* module `star_formation_imf_chabrier`

**subroutine:** `star_formation_imf_initialize_chabrier`
*Description:* Initialize the Chabrier IMF module.
*Code lines:* 37
*Contained by:* module `star_formation_imf_chabrier`
*Modules used:* `input_parameters`

**subroutine:** `star_formation_imf_maximum_mass_chabrier`
*Description:* Register the name of this IMF.
*Code lines:* 13
*Contained by:* module `star_formation_imf_chabrier`

**subroutine:** `star_formation_imf_minimum_mass_chabrier`
*Description:* Register the name of this IMF.
*Code lines:* 13
*Contained by:* module `star_formation_imf_chabrier`

**subroutine:** `star_formation_imf_phi_chabrier`
*Description:* Register the name of this IMF.
*Code lines:* 14
*Contained by:* module `star_formation_imf_chabrier`

**subroutine:** `star_formation_imf_recycled_instantaneous_chabrier`
*Description:* Register the name of this IMF.
*Code lines:* 13
*Contained by:* module `star_formation_imf_chabrier`
subroutine: star_formation_imf_register_chabrier
Description: Register this IMF by incrementing the count and keeping a record of the assigned index.
Code lines: 8
Contained by: module star_formation_imf_chabrier

subroutine: star_formation_imf_register_name_chabrier
Description: Register the name of this IMF.
Code lines: 9
Contained by: module star_formation_imf_chabrier
Modules used: iso_varying_string

subroutine: star_formation_imf_tabulate_chabrier
Description: Register the name of this IMF.
Code lines: 20
Contained by: module star_formation_imf_chabrier
Modules used: tables

subroutine: star_formation_imf_yield_instantaneous_chabrier
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_chabrier

file: star_formation.IMF.F90
Description: Contains a module which implements functionality related to the stellar initial mass function.
Code lines: 1698

module: star_formation_imf
Description: Implements functionality related to the stellar initial mass function.
Code lines: 1678
Contained by: file star_formation.IMF.F90
Modules used: abundances_structure
iso_varying_string
star_formation_imf_chabrier
star_formation_imf_kroupa
star_formation_imf_piecewisepowerlaw
star_formation_imf_scalo
star_formation_imf_yield_instantaneous_chabrier

Used by: function intergalactic_medium_state_internal_update
function stellar_population_luminosity
subroutine stellar_population_properties_rates_instantaneous
subroutine stellar_population_properties_rates_noninstantaneous
subroutine stellar_population_spectra_conroy_initialize_imf

input_parameters
star_formation_imf_baugh2005topheavy
star_formation_imf_kennicutt
star_formation_imf_millerscalo
star_formation_imf_salpeter
star_formation_imf_select_disk_spheroid
star_formation_imf_select_fixed

function radiation_intergalactic_background_internal_update
subroutine stellar_population_properties_rates_instantaneous
subroutine stellar_population_spectra_conroy_initialize_imf
function: cumulative_energy_integrand
   Description: Integrand used in evaluating cumulative energy input.
   Code lines: 25
   Contained by: module star_formation_imf
   Modules used: iso_c_binding stellar_feedback

function: imf_descriptor
   Description: Return a full descriptor for the IMF with the specified index.
   Code lines: 16
   Contained by: module star_formation_imf
   Modules used: galacticus_error

function: imf_energy_input_rate_noninstantaneous
   Description: Returns the energy input rate for a simple stellar population in \((\text{km/s})^2 \text{ Gyr}^{-1}\). The IMF is determined from the given \texttt{starFormationRate} and \texttt{fuelAbundances}. The energy input rate is computed for the given \texttt{age} (in Gyr). The cumulative energy input is computed on a grid of age and metallicity. This is stored to file and will be read back in on subsequent runs. This is useful as computation of the table is relatively slow.
   Code lines: 307
   Contained by: module star_formation_imf
   Modules used: dates_and_times file_utilities
                 fox_dom fox_wxml
                 galacticus_display galacticus_error
                 galacticus_input_paths io_xml
                 iso_c_binding memory_management
                 numerical_constants_astronomical numerical_integration
                 numerical_interpolation numerical_ranges

function: imf_maximum_mass
   Description: Returns the maximum mass in the selected IMF.
   Code lines: 37
   Contained by: module star_formation_imf

function: imf_metal_yield_rate_noninstantaneous
**Description:** Returns the metal yield rate for a simple stellar population, either for the total metallicity or, if `atomIndex` is given, for the specified element. The IMF is determined from the given `starFormationRate` and `fuelAbundances`. The metal yield rate (in fraction of the population’s mass returned to the ISM as new metals per Gyr) is computed for the given `age` (in Gyr). The metal yield is computed on a grid of age and metallicity. This is stored to file and will be read back in on subsequent runs. This is useful as computation of the table is relatively slow.

**Code lines:** 370

**Contained by:** module `star_formation_imf`

**Modules used:**
- `dates_and_times`
- `fox_dom`
- `galacticus_display`
- `galacticus_input_paths`
- `iso_c_binding`
- `memory_management`
- `numerical_integration`
- `numerical_ranges`
- `file_utilities`
- `fox_wxml`
- `galacticus_error`
- `io_xml`
- `kind_numbers`
- `numerical_constants_astronomical`
- `numerical_interpolation`

**Function:** `imf_minimum_mass`
19.1. Program units

**Description:** Returns the minimum mass in the selected IMF.

**Code lines:** 37

**Contained by:** module `star_formation_imf`

**function: imf_name**

**Description:** Return the name of the IMF with the specified index.

**Code lines:** 16

**Contained by:** module `star_formation_imf`

**Modules used:** `galacticus_error`

**function: imf_phi**

**Description:** Returns the IMF, $\Phi(M)$, at mass $M = \text{initialMass}$ for the selected IMF.

**Code lines:** 38

**Contained by:** module `star_formation_imf`

**function: imf_recycled_fractioninstantaneous**

**Description:** Returns a recycled fraction for the IMF suitable for use in the instantaneous recycling approximation.

**Code lines:** 46

**Contained by:** module `star_formation_imf`

**function: imf_recycling_rate_noninstantaneous**

**Description:** Returns the recycling rate for a simple stellar population. The IMF is determined from the given `starFormationRate` and `fuelAbundances`. The recycling rate (in the fraction of the population’s mass returned to the ISM per Gyr) is computed for the given `age` (in Gyr). The recycled fraction is computed on a grid of age and metallicity. This is stored to file and will be read back in on subsequent runs. This is useful as computation of the table is relatively slow.

**Code lines:** 315

**Contained by:** module `star_formation_imf`

**Modules used:**
- `dates_and_times`
- `file_utilities`
- `fox_dom`
- `fox_wxml`
- `galacticus_display`
- `galacticus_error`
- `galacticus_input_paths`
- `io_xml`
- `iso_c_binding`
- `memory_management`
- `numerical_constants_astronomical`
- `numerical_integration`
- `numerical_interpolation`
- `numerical_ranges`

**function: imf_select**

**Description:** Selects an IMF give an input `starFormationRate` and `fuelAbundances`.

**Code lines:** 13

**Contained by:** module `star_formation_imf`

**subroutine: imf_tabulate**

**Description:** Returns a tabulation of the IMF with sufficient resolution to resolve all features.

**Code lines:** 42

**Contained by:** module `star_formation_imf`

**Modules used:** `tables`
function: imf_yield_instantaneous
Description: Returns a yield for the IMF suitable for use in the instantaneous recycling approximation.
Code lines: 46
Contained by: module star_formation_imf

function: metal_yield_integrand
Description: Integrand used in evaluating metal yields.
Code lines: 43
Contained by: module star_formation_imf
Modules used: iso_c_binding, stellar_astrophysics

function: recycled_fraction_integrand
Description: Integrand used in evaluating recycled fractions.
Code lines: 15
Contained by: module star_formation_imf
Modules used: iso_c_binding, stellar_astrophysics

subroutine: star_formation_imf_initialize
Description: Initialize the IMF subsystem.
Code lines: 145
Contained by: module star_formation_imf
Modules used: galacticus_error, memory_management

function: star_is_evolved
Description: Returns true if the specified star is evolved by the given age.
Code lines: 14
Contained by: module star_formation_imf
Modules used: stellar_astrophysics

file: star_formation.IMF.Kennicutt.F90
Description: Contains a module which implements the Kennicutt stellar initial mass function [Kennicutt, 1983].
Code lines: 231

module: star_formation_imf_kennicutt
Description: Implements the Kennicutt stellar initial mass function.
Code lines: 211
Contained by: file star_formation.IMF.Kennicutt.F90
Used by: module star_formation_imf

subroutine: star_formation_imf_initialize_kennicutt
Description: Initialize the Kennicutt IMF module.
Code lines: 42
Contained by: module star_formation_imf_kennicutt
Modules used: input_parameters, star_formation_imf_ppl

subroutine: star_formation_imf_maximum_mass_kennicutt
19.1. Program units

**Description:** Register the name of this IMF.
**Code lines:** 13
**Contained by:** module `star_formation_imf_kennicutt`

**subroutine:** `star_formation_imf_minimum_mass_kennicutt`
**Description:** Register the name of this IMF.
**Code lines:** 13
**Contained by:** module `star_formation_imf_kennicutt`

**subroutine:** `star_formation_imf_phi_kennicutt`
**Description:** Register the name of this IMF.
**Code lines:** 15
**Contained by:** module `star_formation_imf_kennicutt`
**Modules used:** `star_formation_imf_pppl`

**subroutine:** `star_formation_imf_recycled_instantaneous_kennicutt`
**Description:** Register the name of this IMF.
**Code lines:** 13
**Contained by:** module `star_formation_imf_kennicutt`

**subroutine:** `star_formation_imf_register_kennicutt`
**Description:** Register this IMF by incrementing the count and keeping a record of the assigned index.
**Code lines:** 8
**Contained by:** module `star_formation_imf_kennicutt`

**subroutine:** `star_formation_imf_register_name_kennicutt`
**Description:** Register the name of this IMF.
**Code lines:** 9
**Contained by:** module `star_formation_imf_kennicutt`
**Modules used:** `iso_varying_string`

**subroutine:** `star_formation_imf_tabulate_kennicutt`
**Description:** Register the name of this IMF.
**Code lines:** 21
**Contained by:** module `star_formation_imf_kennicutt`
**Modules used:** `star_formation_imf_pppl` tables

**subroutine:** `star_formation_imf_yield_instantaneous_kennicutt`
**Description:** Register the name of this IMF.
**Code lines:** 13
**Contained by:** module `star_formation_imf_kennicutt`

**file:** `star_formation.IMF.Kroupa.F90`
**Description:** Contains a module which implements the Kroupa stellar initial mass function [Kroupa, 2001].
**Code lines:** 231

**module:** `star_formation_imf_kroupa`
**Description:** Implements the Kroupa stellar initial mass function.
19. Source Code Documentation

**Code lines:** 211  
**Contained by:** file `star_formation.IMF.Kroupa.F90`  
**Used by:** module `star_formation_imf`

**subroutine:** `star_formation_imf_initialize_kroupa`  
**Description:** Initialize the Kroupa IMF module.  
**Code lines:** 42  
**Contained by:** module `star_formation_imf_kroupa`  
**Modules used:** `input_parameters` `star_formation_imf_ppl`

**subroutine:** `star_formation_imf_maximum_mass_kroupa`  
**Description:** Register the name of this IMF.  
**Code lines:** 13  
**Contained by:** module `star_formation_imf_kroupa`

**subroutine:** `star_formation_imf_minimum_mass_kroupa`  
**Description:** Register the name of this IMF.  
**Code lines:** 13  
**Contained by:** module `star_formation_imf_kroupa`

**subroutine:** `star_formation_imf_phi_kroupa`  
**Description:** Register the name of this IMF.  
**Code lines:** 15  
**Contained by:** module `star_formation_imf_kroupa`  
**Modules used:** `star_formation_imf_ppl`

**subroutine:** `star_formation_imf_recycled_instantaneous_kroupa`  
**Description:** Register the name of this IMF.  
**Code lines:** 13  
**Contained by:** module `star_formation_imf_kroupa`

**subroutine:** `star_formation_imf_register_kroupa`  
**Description:** Register this IMF by incrementing the count and keeping a record of the assigned index.  
**Code lines:** 8  
**Contained by:** module `star_formation_imf_kroupa`

**subroutine:** `star_formation_imf_register_name_kroupa`  
**Description:** Register the name of this IMF.  
**Code lines:** 9  
**Contained by:** module `star_formation_imf_kroupa`  
**Modules used:** `iso_varying_string`

**subroutine:** `star_formation_imf_tabulate_kroupa`  
**Description:** Register the name of this IMF.  
**Code lines:** 21  
**Contained by:** module `star_formation_imf_kroupa`  
**Modules used:** `star_formation_imf_ppl` `tables`
19.1. Program units

subroutine: star_formation_imf_yield_instantaneous_kroupa
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_kroupa

code: star_formation.IMF.Miller-Scalo.F90
Description: Contains a module which implements the Miller-Scalo stellar initial mass function [Miller and Scalo, 1979].
Code lines: 231

module: star_formation_imf_millerscalo
Description: Implements the MillerScalo stellar initial mass function.
Code lines: 211
Contained by: file star_formation.IMF.Miller-Scalo.F90
Used by: module star_formation_imf

subroutine: star_formation_imf_initialize_millerscalo
Description: Initialize the MillerScalo IMF module.
Code lines: 42
Contained by: module star_formation_imf_millerscalo
Modules used: input_parameters star_formation_imf_ppl

subroutine: star_formation_imf_maximum_mass_millerscalo
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_millerscalo

subroutine: star_formation_imf_minimum_mass_millerscalo
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_millerscalo

subroutine: star_formation_imf_phi_millerscalo
Description: Register the name of this IMF.
Code lines: 15
Contained by: module star_formation_imf_millerscalo
Modules used: star_formation_imf_ppl

subroutine: star_formation_imf_recycled_instantaneous_millerscalo
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_millerscalo

subroutine: star_formation_imf_register_millerscalo
Description: Register this IMF by incrementing the count and keeping a record of the assigned index.
Code lines: 8
Contained by: module star_formation_imf_millerscalo
subroutine: star_formation_imf_register_name_millerscalo
Description: Register the name of this IMF.
Code lines: 9
Contained by: module star_formation_imf_millerscalo
Modules used: iso_varying_string

subroutine: star_formation_imf_tabulate_millerscalo
Description: Register the name of this IMF.
Code lines: 21
Contained by: module star_formation_imf_millerscalo
Modules used: star_formation_imf_ppl    tables

subroutine: star_formation_imf_yield_instantaneous_millerscalo
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_millerscalo

file: star_formation.IMF.Salpeter.F90
Description: Contains a module which implements the Salpeter stellar initial mass function [Salpeter, 1955].
Code lines: 231

module: star_formation_imf_salpeter
Description: Implements the Salpeter stellar initial mass function.
Code lines: 211
Contained by: file star_formation.IMF.Salpeter.F90
Used by: module star_formation_imf

subroutine: star_formation_imf_initialize_salpeter
Description: Initialize the Salpeter IMF module.
Code lines: 42
Contained by: module star_formation_imf_salpeter
Modules used: input_parameters star_formation_imf_ppl

subroutine: star_formation_imf_maximum_mass_salpeter
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_salpeter

subroutine: star_formation_imf_minimum_mass_salpeter
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_salpeter

subroutine: star_formation_imf_phi_salpeter
Description: Register the name of this IMF.
Code lines: 15
Contained by: module star_formation_imf_salpeter
19.1. Program units

Modules used: star_formation_imf_ppl

subroutine: star_formation_imf_recycled_instantaneous_salpeter
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_salpeter

subroutine: star_formation_imf_register_name_salpeter
Description: Register the name of this IMF.
Code lines: 9
Contained by: module star_formation_imf_salpeter
Modules used: iso_varying_string

subroutine: star_formation_imf_register_salpeter
Description: Register this IMF by incrementing the count and keeping a record of the assigned index.
Code lines: 8
Contained by: module star_formation_imf_salpeter

subroutine: star_formation_imf_tabulate_salpeter
Description: Register the name of this IMF.
Code lines: 21
Contained by: module star_formation_imf_salpeter
Modules used: star_formation_imf_ppl

subroutine: star_formation_imf_yield_instantaneous_salpeter
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_salpeter

file: star_formation.IMF.Scalo.F90
Description: Contains a module which implements the Scalo stellar initial mass function [Scalo, 1986].
Code lines: 231

module: star_formation_imf_scalo
Description: Implements the Scalo stellar initial mass function.
Code lines: 211
Contained by: file star_formation.IMF.Scalo.F90
Used by: module star_formation_imf

subroutine: star_formation_imf_initialize_scalo
Description: Initialize the Scalo IMF module.
Code lines: 42
Contained by: module star_formation_imf_scalo
Modules used: input_parameters

subroutine: star_formation_imf_maximum_mass_scalo
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_scalo
subroutine: star_formation_imf_minimum_mass_scalo
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_scalo

subroutine: star_formation_imf_phi_scalo
Description: Register the name of this IMF.
Code lines: 15
Contained by: module star_formation_imf_scalo
Modules used: star_formation_imf_ppl

subroutine: star_formation_imf_recycled_instantaneous_scalo
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_scalo

subroutine: star_formation_imf_register_name_scalo
Description: Register the name of this IMF.
Code lines: 9
Contained by: module star_formation_imf_scalo
Modules used: iso_varying_string

subroutine: star_formation_imf_register_scalo
Description: Register this IMF by incrementing the count and keeping a record of the assigned index.
Code lines: 8
Contained by: module star_formation_imf_scalo

subroutine: star_formation_imf_tabulate_scalo
Description: Register the name of this IMF.
Code lines: 21
Contained by: module star_formation_imf_scalo
Modules used: star_formation_imf_ppl tables

subroutine: star_formation_imf_yield_instantaneous_scalo
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_scalo

file: star_formation.IMF.piecewise_power_law.F90
Description: Contains a module which implements an arbitrary piecewise power-law stellar initial mass function.
Code lines: 300

module: star_formation_imf_piecewisepowerlaw
Description: Implements an arbitrary piecewise power-law stellar initial mass function.
Code lines: 280
Contained by: file star_formation.IMF.piecewise_power_law.F90
Used by: module star_formation_imf
subroutine: star_formation_imf_initialize_piecewisepowerlaw
Description: Initialize the PiecewisePowerLaw IMF module.
Code lines: 98
Contained by: module star_formation_imf_piecewisepowerlaw
Modules used: galacticus_error input_parameters
                  memory_management star_formation_imf_ppl

subroutine: star_formation_imf_maximum_mass_piecewisepowerlaw
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_piecewisepowerlaw

subroutine: star_formation_imf_minimum_mass_piecewisepowerlaw
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_piecewisepowerlaw

subroutine: star_formation_imf_phi_piecewisepowerlaw
Description: Register the name of this IMF.
Code lines: 15
Contained by: module star_formation_imf_piecewisepowerlaw
Modules used: star_formation_imf_ppl

subroutine: star_formation_imf_recycled_instantaneous_piecewisepowerlaw
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_piecewisepowerlaw

subroutine: star_formation_imf_register_name_piecewisepowerlaw
Description: Register this IMF by incrementing the count and keeping a record of the assigned index.
Code lines: 22
Contained by: module star_formation_imf_piecewisepowerlaw
Modules used: iso_varying_string

subroutine: star_formation_imf_register_piecewisepowerlaw
Description: Register this IMF by incrementing the count and keeping a record of the assigned index.
Code lines: 8
Contained by: module star_formation_imf_piecewisepowerlaw

subroutine: star_formation_imf_tabulate_piecewisepowerlaw
Description: Register the name of this IMF.
Code lines: 21
Contained by: module star_formation_imf_piecewisepowerlaw
Modules used: star_formation_imf_ppl tables

subroutine: star_formation_imf_yield_instantaneous_piecewisepowerlaw
Description: Register the name of this IMF.
Code lines: 13
19. Source Code Documentation

**Contained by:** module `star_formation_imf_piecewisepowerlaw`

**file:** `star_formation.IMF.select.diskSpheroid.F90`

*Description:* Contains a module which implements selection of stellar IMFs with one IMF for disks and another for spheroids.

*Code lines:* 99

**module:** `star_formation_imf_select_disk_spheroid`

*Description:* Implements selection of stellar IMFs with one IMF for disks and another for spheroids.

*Code lines:* 79

**Contained by:** file `star_formation.IMF.select.diskSpheroid.F90`

**Used by:** module `star_formation_imf`

**function:** `imf_select_disk_spheroid`

*Description:* Return our selection of stellar initial mass function.

*Code lines:* 20

**Contained by:** module `star_formation_imf_select_disk_spheroid`

**Modules used:**
- `abundances_structure`
- `galactic_structure_options`
- `galacticus_error`

**subroutine:** `imf_select_disk_spheroid_initialize`

*Description:* Initializes the “diskSpheroid” IMF selection module.

*Code lines:* 41

**Contained by:** module `star_formation_imf_select_disk_spheroid`

**Modules used:**
- `input_parameters`
- `iso_varying_string`
- `star_formation_imf_utilities`

**file:** `star_formation.IMF.select.fixed.F90`

*Description:* Contains a module which implements a fixed choice of stellar initial mass function.

*Code lines:* 76

**module:** `star_formation_imf_select_fixed`

*Description:* Implements a fixed choice of stellar initial mass function.

*Code lines:* 56

**Contained by:** file `star_formation.IMF.select.fixed.F90`

**Used by:** module `star_formation_imf`

**function:** `imf_select_fixed`

*Description:* Return our selection of stellar initial mass function.

*Code lines:* 10

**Contained by:** module `star_formation_imf_select_fixed`

**Modules used:**
- `abundances_structure`

**subroutine:** `imf_select_fixed_initialize`

*Description:* Initializes the “fixed” IMF selection module.

*Code lines:* 28

**Contained by:** module `star_formation_imf_select_fixed`

**Modules used:**
- `input_parameters`
- `iso_varying_string`
- `star_formation_imf_utilities`
**module:** star\_formation\_imf\_utilities

*Description:* Contains useful utilities required by the IMF subsystem.

*Code lines:* 24

*Contained by:* file star\_formation.IMF.utilities.F90

*Used by:* subroutine imf\_select\_disk\_spheroid\_initialize

function: imf\_index\_lookup

*Description:* Returns the internal index of a stellar initial mass function specified by name via imfSelection.

*Code lines:* 14

*Contained by:* module star\_formation\_imf\_utilities

*Modules used:* galacticus\_error iso\_varying\_string

**file:** star\_formation.IMF.utilities.piecewise\_power\_laws.F90

*Description:* Contains a module which implements calculations of piecewise power-law initial mass functions.

*Code lines:* 102

**module:** star\_formation\_imf\_ppl

*Description:* Implements calculations of piecewise power-law initial mass functions.

*Code lines:* 82

*Contained by:* file star\_formation.IMF.utilities.piecewise\_power\_laws.F90

*Used by:* subroutine star\_formation\_imf\_initialize\_baugh2005topheavy subroutine star\_formation\_imf\_phi\_baugh2005topheavy subroutine star\_formation\_imf\_tabulate\_baugh2005topheavy subroutine star\_formation\_imf\_phi\_tabulate\_baugh2005topheavy subroutine star\_formation\_imf\_phi\_kennicutt subroutine star\_formation\_imf\_tabulate\_kennicutt subroutine star\_formation\_imf\_phi\_tabulate\_kennicutt subroutine star\_formation\_imf\_initialize\_kroupa subroutine star\_formation\_imf\_phi\_initialize\_kroupa subroutine star\_formation\_imf\_tabulate\_kroupa subroutine star\_formation\_imf\_phi\_tabulate\_kroupa subroutine star\_formation\_imf\_initialize\_millerscalo subroutine star\_formation\_imf\_phi\_initialize\_millerscalo subroutine star\_formation\_imf\_tabulate\_millerscalo subroutine star\_formation\_imf\_phi\_tabulate\_millerscalo subroutine star\_formation\_imf\_initialize\_salpeter subroutine star\_formation\_imf\_phi\_initialize\_salpeter subroutine star\_formation\_imf\_tabulate\_salpeter subroutine star\_formation\_imf\_phi\_tabulate\_salpeter subroutine star\_formation\_imf\_phi\_initialize\_piecewisepowerlaw subroutine star\_formation\_imf\_phi\_initialize\_piecewisepowerlaw subroutine star\_formation\_imf\_phi\_tabulate\_piecewisepowerlaw subroutine star\_formation\_imf\_phi\_tabulate\_piecewisepowerlaw
19. Source Code Documentation

subroutine star_formation_imf_-tabulate_piecewisepowerlaw

subroutine: piecewise_power_law_imf_normalize
Description: Computes normalizations for the pieces of a piecewise power-law IMF such that the IMF is continuous and normalized to unit stellar mass.
Code lines: 32
Contained by: module star_formation_imf_ppl

interface: piecewise_power_law_imf_phi
Code lines: 3
Contained by: module star_formation_imf_ppl

function: piecewise_power_law_imf_phi_array
Description: Returns the IMF at given mass().
Code lines: 12
Contained by: module star_formation_imf_ppl

function: piecewise_power_law_imf_phi_scalar
Description: Returns the IMF at given mass.
Code lines: 18
Contained by: module star_formation_imf_ppl

file: star_formation.feedback.disks.Creasey2012.F90
Description: Contains a module which implements the Creasey et al. [2012] model for star formation feedback in galactic disks.
Code lines: 162

module: star_formation_feedback_disks_creasey2012
Description: Implements the Creasey et al. [2012] model for star formation feedback in galactic disks.
Code lines: 142
Contained by: file star_formation.feedback.disks.Creasey2012.F90
Modules used: galacticus_nodes
Used by: subroutine star_formation_feedback_-disks_initialize

function: star_formation_feedback_disk_outflow_rate_creasey2012
**Description:** Returns the outflow rate (in $M_\odot\text{Gyr}^{-1}$) for star formation in the galactic disk of `thisNode` using the model of Creasey et al. [2012]. The outflow rate is given by

$$M_{\text{outflow}} = \int_0^\infty \beta_0 \Sigma_{g,1}^{-1}(r)f_g(r)\Sigma_\star(r)2\pi r dr,$$

(19.30)

where $\Sigma_{g,1}(r)$ is the surface density of gas in units of $M_\odot\text{pc}^{-2}$, $f_g(r)$ is the gas fraction, $\Sigma_\star(r)$ is the surface density of star formation rate, $\beta_0$ = [starFormationFeedbackDisksCreasy2012Beta0], $\mu$ = [starFormationFeedbackDisksCreasy2012Mu], and $\nu$ = [starFormationFeedbackDisksCreasy2012Nu].

**Code lines:** 43

**Contained by:** module `star_formation_feedback_disks_creasey2012`

**Modules used:** `iso_c_binding`, `numerical_constants_math`, `numerical_integration`, `stellar_feedback`
Description: Integrand function for the “Creasey et al. (2012)” supernovae feedback calculation.

Code lines: 26

Contained by: module star_formation_feedback_disks_creasey2012

Modules used: galactic_structure_options  galactic_structure_surface_densities
iso_c_binding  numerical_constants_prefixes
star_formation_rate_surface_density_disks

subroutine: star_formation_feedback_disks_creasey2012_initialize
Description: Initializes the “Creasey et al. (2012)” disk star formation feedback module.

Code lines: 49

Contained by: module star_formation_feedback_disks_creasey2012

Modules used: input_parameters  iso_varying_string

file: star_formation.feedback.disks.F90
Description: Contains a module which implements calculations of feedback from star formation in disks.

Code lines: 122

module: star_formation_feedback_disks
Description: Implements calculations of feedback from star formation in disks.

Code lines: 102

Contained by: file star_formation.feedback.disks.F90

Modules used: galacticus_nodes  iso_varying_string

Used by: subroutine node_component_disk_exponential_rate_compute  subroutine node_component_disk_very_simple_rate_compute

function: star_formation_feedback_disk_outflow_rate
Description: Returns the outflow rate due to star formation in the disk component of thisNode.

Code lines: 13

Contained by: module star_formation_feedback_disks

interface: star_formation_feedback_disk_outflow_rate_template

Code lines: 6

Contained by: module star_formation_feedback_disks

function: star_formation_feedback_disk_outflow_rate_template

Code lines: 4

Contained by: interface star_formation_feedback_disk_outflow_rate_template

subroutine: star_formation_feedback_disks_initialize
Description: Initialize the disk star formation feedback module.

Code lines: 59

Contained by: module star_formation_feedback_disks

Modules used: galacticus_error  input_parameters
star_formation_feedback_disks_creasey2012  star_formation_feedback_disks_fixed
star_formation_feedback_disks_halo_scaling  star_formation_feedback_disks_power_law
file: star_formation.feedback.disks.fixed.F90
Description: Contains a module which implements a fixed outflow rate due to star formation feedback in galactic disks.
Code lines: 74

module: star_formation_feedback_disks_fixed
Description: Implements a fixed outflow rate due to star formation feedback in galactic disks.
Code lines: 54
Contained by: file star_formation.feedback.disks.fixed.F90
Modules used: galacticus_nodes
Used by: subroutine star_formation_feedback_disks_fixed_initialize

function: star_formation_feedback_disk_outflow_rate_fixed
Description: Returns the outflow rate (in $M_\odot \text{Gyr}^{-1}$) for star formation in the galactic disk of thisNode. Assumes a fixed ratio of outflow rate to star formation rate.
Code lines: 10
Contained by: module star_formation_feedback_disks_fixed
Modules used: stellar_feedback

subroutine: star_formation_feedback_disks_fixed_initialize
Description: Initializes the “fixed” disk star formation feedback module.
Code lines: 25
Contained by: module star_formation_feedback_disks_fixed
Modules used: input_parameters iso_varying_string

file: star_formation.feedback.disks.halo_scaling.F90
Description: Contains a module which implements an outflow rate due to star formation feedback in galactic disks that scales with halo virial velocity and redshift.
Code lines: 138

module: star_formation_feedback_disks_halo_scaling
Description: Implements an outflow rate due to star formation feedback in galactic disks that scales with halo virial velocity and redshift.
Code lines: 117
Contained by: file star_formation.feedback.disks.halo_scaling.F90
Modules used: galacticus_nodes
Used by: subroutine star_formation_feedback_disks_halo_scaling_initialize

function: star_formation_feedback_disk_outflow_rate_halo_scaling
Description: Returns the outflow rate (in $M_\odot \text{Gyr}^{-1}$) for star formation in the galactic disk of thisNode.
Code lines: 41
Contained by: module star_formation_feedback_disks_halo_scaling
Modules used: cosmology_functions dark_matter_halo_scales

subroutine: star_formation_feedback_disks_halo_scaling_initialize
Description: Initializes the “halo scaling” disk star formation feedback module.
Code lines: 53
19. Source Code Documentation

**Contained by:** module `star_formation_feedback_disks_halo_scaling`

**Modules used:** `input_parameters`  `iso_varying_string`

**stellar_feedback**

**file:** `star_formation.feedback.disks.power_law.F90`

**Description:** Contains a module which implements a power-law outflow rate due to star formation feedback in galactic disks.

**Code lines:** 107

**module:** `star_formation_feedback_disks_power_law`

**Description:** Implements a power-law outflow rate due to star formation feedback in galactic disks.

**Code lines:** 87

**Contained by:** file `star_formation.feedback.disks.power_law.F90`

**Used by:** subroutine `star_formation_feedback_disks_initialize`

**function:** `star_formation_feedback_disk_outflow_rate_power_law`

**Description:** Returns the outflow rate (in $M_\odot \, \text{Gyr}^{-1}$) for star formation in the galactic disk of `thisNode`. The outflow rate is given by

$$
\dot{M}_{\text{outflow}} = \left( \frac{V_{\text{disk, outflow}}}{V_{\text{disk}}} \right)^{\alpha_{\text{disk, outflow}}},
$$

where $V_{\text{disk, outflow}} (= \text{diskOutflowVelocity})$ is the velocity scale at which outflow rate equals star formation rate and $\alpha_{\text{disk, outflow}} (= \text{diskOutflowExponent})$ controls the scaling with velocity. Note that the velocity $V_{\text{disk}}$ is whatever characteristic value returned by the disk method. This scaling is functionally similar to that adopted by Cole et al. [2000], but that they specifically used the circular velocity at half-mass radius.

**Code lines:** 32

**Contained by:** module `star_formation_feedback_disks_power_law`

**Modules used:** `galacticus_nodes`  `stellar_feedback`

**subroutine:** `star_formation_feedback_disks_power_law_initialize`

**Description:** Initializes the “power law” disk star formation feedback module.

**Code lines:** 37

**Contained by:** module `star_formation_feedback_disks_power_law`

**Modules used:** `input_parameters`  `iso_varying_string`

**file:** `star_formation.feedback.spheroids.F90`

**Description:** Contains a module which implements calculations of feedback from star formation in spheroids.

**Code lines:** 116

**module:** `star_formation_feedback_spheroids`

**Description:** Implements calculations of feedback from star formation in spheroids.

**Code lines:** 96

**Contained by:** file `star_formation.feedback.spheroids.F90`

**Modules used:** `galacticus_nodes`  `iso_varying_string`

**Used by:** subroutine `node_component_spheroid_standard_rate_compute`
function: star_formation_feedback_spheroid_outflow_rate
Description: Returns the outflow rate due to star formation in the spheroid component of thisNode.
Code lines: 13
Contained by: module star_formation_feedback_spheroids

interface: star_formation_feedback_spheroid_outflow_rate_template
Code lines: 6
Contained by: module star_formation_feedback_spheroids

function: star_formation_feedback_spheroid_outflow_rate_template
Code lines: 4
Contained by: interface star_formation_feedback_spheroid_outflow_rate_template

subroutine: star_formation_feedback_spheroids_initialize
Description: Initialize the spheroid star formation feedback module.
Code lines: 53
Contained by: module star_formation_feedback_spheroids
Modules used: galacticus_error input_parameters
star_formation_feedback_spheroids_-
power_law

file: star_formation.feedback.spheroids.power_law.F90
Description: Contains a module which implements a power-law outflow rate due to star formation feedback in galactic spheroids.
Code lines: 105

module: star_formation_feedback_spheroids_power_law
Description: Implements a power-law outflow rate due to star formation feedback in galactic spheroids.
Code lines: 85
Contained by: file star_formation.feedback.spheroids.power_law.F90
Used by: subroutine star_formation_feedback_spheroids_initialize

function: star_formation_feedback_spheroid_outflow_rate_power_law
Description: Returns the outflow rate (in \(M_\odot\) Gyr\(^{-1}\)) for star formation in the galactic spheroid of thisNode. The outflow rate is given by

\[
\dot{M}_{\text{outflow}} = \left( \frac{V_{\text{spheroid,outflow}}}{V_{\text{spheroid}}} \right)^{\alpha_{\text{spheroid,outflow}}},
\]

where \(V_{\text{spheroid,outflow}}\) is the velocity scale at which outflow rate equals star formation rate and \(\alpha_{\text{spheroid,outflow}}\) controls the scaling with velocity. Note that the velocity \(V_{\text{spheroid}}\) is whatever characteristic value returned by the spheroid method. This scaling is functionally similar to that adopted by Cole et al. [2000], but that they specifically used the circular velocity at half-mass radius.
Code lines: 30
Contained by: module star_formation_feedback_spheroids_power_law
Modules used: galacticus_nodes stellar_feedback

subroutine: star_formation_feedback_spheroids_power_law_initialize
19. Source Code Documentation

**Description:** Initializes the “power law” spheroid star formation feedback module.

**Code lines:** 37

**Contained by:** module `star_formation_feedback_spheroids_power_law`

**Modules used:** `input_parameters` `iso_varying_string`

**file:** `star_formation.feedback_expulsion.disks.F90`

**Description:** Contains a module which implements calculations of expulsive feedback from star formation in disks.

**Code lines:** 119

**module:** `star_formation_feedback_expulsion_disks`

**Description:** Implements calculations of expulsive feedback from star formation in disks.

**Code lines:** 99

**Contained by:** file `star_formation.feedback_expulsion.disks.F90`

**Modules used:** `galacticus_nodes` `iso_varying_string`

**Used by:** subroutine `node_component_disk_exponential_rate_compute`

**function:** `star_formation_expulsive_feedback_disk_outflow_rate`

**Description:** Returns the expulsive outflow rate due to star formation in the disk component of `thisNode`.

**Code lines:** 13

**Contained by:** module `star_formation_feedback_expulsion_disks`

**subroutine:** `star_formation_expulsive_feedback_disks_initialize`

**Description:** Initialize the disk star formation expulsive feedback module.

**Code lines:** 56

**Contained by:** module `star_formation_feedback_expulsion_disks`

**Modules used:** `galacticus_error` `input_parameters` `star_formation_expulsive_feedback_disks_null` `star_formation_expulsive_feedback_disks_superwind`

**file:** `star_formation.feedback_expulsion.disks.null.F90`

**Description:** Contains a module which implements a null expulsive outflow rate in galactic disks.

**Code lines:** 55

**module:** `star_formation_feedback_expulsion_disks_null`

**Description:** Implements a null expulsive outflow rate in galactic disks.

**Code lines:** 35

**Contained by:** file `star_formation.feedback_expulsion.disks.null.F90`

**Used by:** subroutine `star_formation_feedback_disks_null_initialize`

**function:** `star_formation_expulsive_feedback_disk_outflow_rate_null`

**Description:** Implements a null expulsive outflow rate for disks.

**Code lines:** 10

**Contained by:** module `star_formation_feedback_expulsion_disks_null`

**Modules used:** `galacticus_nodes`

**subroutine:** `star_formation_feedback_disks_null_initialize`
19.1. Program units

Description: Initializes the “null” disk star formation expulsive feedback module.
Code lines: 10
Contained by: module `star_formation_expulsive_feedback_disks_null`
Modules used: `iso_varying_string`

file: `star_formation.feedback_expulsion.disks.superwind.F90`
Description: Contains a module which implements a “superwind” expulsive outflow rate (as in [Baugh et al., 2005]) due to star formation feedback in galactic disks.
Code lines: 110

module: `star_formation_expulsive_feedback_disks_superwind`
Description: Implements a “superwind” outflow rate (as in [Baugh et al., 2005]) due to star formation feedback in galactic disks.
Code lines: 89
Contained by: file `star_formation.feedback_expulsion.disks.superwind.F90`
Used by: subroutine `star_formation_expulsive_feedback_disks_initialize`

function: `star_formation_expulsive_feedback_disk_outflow_rate_sw`
Description: Returns the expulsive outflow rate (in $M_\odot$ Gyr$^{-1}$) for star formation in the galactic disk of thisNode. The outflow rate is given by

$$M_{\text{outflow}} = f_{SW,0} \begin{cases} 
1 & \text{if } V_{\text{disk}} < V_{\text{disk,SW}} \\
(V_{\text{disk,SW}}/V_{\text{disk}})^2 & \text{if } V_{\text{disk}} \geq V_{\text{disk,SW}} 
\end{cases} \quad (19.33)$$

where $V_{\text{disk,SW}} = [\text{diskSuperwindVelocity}]$ and $f_{SW,0} = [\text{diskSuperwindMassLoading}]$.
Note that the velocity $V_{\text{disk}}$ is whatever characteristic value returned by the disk method. This scaling is functionally similar to that adopted by Cole et al. [2000] and Baugh et al. [2005], except that they specifically used the circular velocity at half-mass radius.

Code lines: 33
Contained by: module `star_formation_expulsive_feedback_disks_superwind`
Modules used: `galacticus_nodes` `stellar_feedback`

subroutine: `star_formation_expulsive_feedback_disks_sw_initialize`
Description: Initializes the ‘superwind’ disk star formation expulsive feedback module.
Code lines: 37
Contained by: module `star_formation_expulsive_feedback_disks_superwind`
Modules used: `input_parameters` `iso_varying_string`

file: `star_formation.feedback_expulsion.spheroids.F90`
Description: Contains a module which implements calculations of expulsive feedback from star formation in spheroids.
Code lines: 118

module: `star_formation_feedback_expulsion_spheroids`
Description: Implements calculations of expulsive feedback from star formation in spheroids.
Code lines: 98
Contained by: file `star_formation.feedback_expulsion.spheroids.F90`
Modules used: `galacticus_nodes` `iso_varying_string`
19. Source Code Documentation

**Used by:** subroutine `node_component_spheroid-standard_rate_compute`

**function:** `star_formation_expulsive_feedback_spheroid_outflow_rate`

*Description:* Returns the expulsive outflow rate due to star formation in the spheroid component of thisNode.

*Code lines:* 13

*Contained by:* module `star_formation_feedback_expulsion_spheroids`

**subroutine:** `star_formation_expulsive_feedback_spheroids_initialize`

*Description:* Initialize the spheroid star formation expulsive feedback module.

*Code lines:* 55

*Contained by:* module `star_formation_feedback_expulsion_spheroids`

**Modules used:**
- `galacticus_error`
- `input_parameters`
- `star_formation_expulsive_feedback_spheroids`
- `star_formation_expulsive_feedback_spheroids_null`
- `star_formation_expulsive_feedback_spheroids_superwind`

**file:** `star_formation.feedback_expulsion.spheroids.null.F90`

*Description:* Contains a module which implements a null expulsive outflow rate in galactic spheroids.

*Code lines:* 55

**module:** `star_formation_expulsive_feedback_spheroids_null`

*Description:* Implements a null expulsive outflow rate in galactic spheroids.

*Code lines:* 35

*Contained by:* file `star_formation.feedback_expulsion.spheroids.null.F90`

*Used by:* subroutine `star_formation_expulsive_feedback_spheroids_null_initialize`

**function:** `star_formation_expulsive_feedback_spheroid_outflow_rate_null`

*Description:* Implements a null expulsive outflow rate for spheroids.

*Code lines:* 10

*Contained by:* module `star_formation_expulsive_feedback_spheroids_null`

*Modules used:* `galacticus_nodes`

**subroutine:** `star_formation_expulsive_feedback_spheroids_null_initialize`

*Description:* Initializes the “null” spheroid star formation expulsive feedback module.

*Code lines:* 10

*Contained by:* module `star_formation_expulsive_feedback_spheroids_null`

*Modules used:* `iso_varying_string`

**file:** `star_formation.feedback_expulsion.spheroids.superwind.F90`

*Description:* Contains a module which implements a “superwind” expulsive outflow rate (as in [Baugh et al., 2005]) due to star formation feedback in galactic spheroids.

*Code lines:* 108

**module:** `star_formation_expulsive_feedback_spheroids_superwind`

*Description:* Implements a “superwind” outflow rate (as in [Baugh et al., 2005]) due to star formation feedback in galactic spheroids.

*Code lines:* 87

*Contained by:* file `star_formation.feedback_expulsion.spheroids.superwind.F90`
19.1. Program units

Used by: subroutine star_formation_expulsive_feedback_spheroids_initialize

function: star_formation_expulsive_feedback_spheroid_outflow_rate_sw

Description: Returns the expulsive outflow rate (in $M_{\odot}$ Gyr$^{-1}$) for star formation in the galactic spheroid of thisNode. The outflow rate is given by

$$\dot{M}_{\text{outflow}} = f_{SW,0} \left\{ \begin{array}{ll} 1 & \text{if } V_{\text{spheroid}} < V_{\text{spheroid},SW} \\ (V_{\text{spheroid},SW}/V_{\text{spheroid}})^2 & \text{if } V_{\text{spheroid}} \geq V_{\text{spheroid},SW} \end{array} \right., \quad (19.34)$$

where $V_{\text{spheroid},SW} = [\text{spheroidSuperwindVelocity}]$ and $f_{SW,0} = [\text{spheroidSuperwindMassLoading}]$. Note that the velocity $V_{\text{spheroid}}$ is whatever characteristic value returned by the spheroid method. This scaling is functionally similar to that adopted by Cole et al. [2000] and Baugh et al. [2005], except that they specifically used the circular velocity at half-mass radius.

Code lines: 31
Contained by: module star_formation_expulsive_feedback_spheroids_superwind
Modules used: galacticus_nodes stellar_feedback

subroutine: star_formation_expulsive_feedback_spheroids_sw_initialize

Description: Initializes the “superwind” spheroid star formation expulsive feedback module.

Code lines: 37
Contained by: module star_formation_expulsive_feedback_spheroids_superwind
Modules used: input_parameters iso_varying_string

file: star_formation.rate_surface_density.disks.Blitz-Rosolowsky.F90

Description: Contains a module which implements the Blitz and Rosolowsky [2006] star formation rate surface density law for galactic disks.

Code lines: 220

module: star_formation_rate_surface_density_disks_br

Description: Implements the Blitz and Rosolowsky [2006] star formation rate surface density law for galactic disks.

Code lines: 200
Contained by: file star_formation.rate_surface_density.disks.Blitz-Rosolowsky.F90
Modules used: galacticus_nodes kind_numbers
Used by: subroutine galacticus_calculations_reset subroutine star_formation_rate_surface_density_disks_br_initialize

function: star_formation_rate_surface_density_disks_br

Description: Returns the star formation rate surface density (in $M_{\odot}$ Gyr$^{-1}$ Mpc$^{-2}$) for star formation in the galactic disk of thisNode. The disk is assumed to obey the Blitz and Rosolowsky [2006] star formation rule.

Code lines: 53
Contained by: module star_formation_rate_surface_density_disks_br
Modules used: abundances_structure galactic_structure_options galactic_structure_surface_densities numerical_constants_math numerical_constants_physical

subroutine: star_formation_rate_surface_density_disks_br_initialize
19. Source Code Documentation

**Description:** Initializes the “extended Schmidt” disk star formation rate surface density.

**Code lines:** 105

**Contained by:** module `star_formation_rate_surface_density_disks_br`

**Modules used:**
- `galacticus_error`
- `input_parameters`
- `iso_varying_string`
- `numerical_constants_astronomical`
- `numerical_constants_physical`

**subroutine:** `star_formation_rate_surface_density_disks_br_reset`

**Description:** Reset the extended Schmidt relation calculation.

**Code lines:** 8

**Contained by:** module `star_formation_rate_surface_density_disks_br`

**file:** `star_formation.rate_surface_density.disks.F90`

**Description:** Contains a module which implements calculations of star formation rate surface densities for galactic disks.

**Code lines:** 114

**module:** `star_formation_rate_surface_density_disks`

**Description:** Implements calculations of star formation rate surface densities for galactic disks.

**Code lines:** 94

**Contained by:** file `star_formation.rate_surface_density.disks.F90`

**Modules used:**
- `galacticus_nodes`
- `iso_varying_string`

**Used by:**
- function `star_formation_feedback_disk_outflow_rate_cressey2012_integrand`
- function `star_formation_rate_integrand_surface_density`

**function:** `star_formation_rate_surface_density_disk`

**Description:** Returns the star formation rate surface density (in \(M_\odot\) Gyr\(^{-1}\) Mpc\(^{-2}\)) in the disk component of thisNode at the given radius.

**Code lines:** 12

**Contained by:** module `star_formation_rate_surface_density_disks`

**subroutine:** `star_formation_rate_surface_density_disks_initialize`

**Description:** Initialize the disk star formation rate surface density module.

**Code lines:** 59

**Contained by:** module `star_formation_rate_surface_density_disks`

**Modules used:**
- `galacticus_error`
- `input_parameters`
- `star_formation_rate_surface_density_disks_br`
- `star_formation_rate_surface_density_disks_exschmidt`
- `star_formation_rate_surface_density_disks_kmt09`
- `star_formation_rate_surface_density_disks_ks`

**file:** `star_formation.rate_surface_density.disks.KMT09.F90`

**Description:** Contains a module which implements the Krumholz et al. [2009] star formation rate surface density law for galactic disks.

**Code lines:** 228

**module:** `star_formation_rate_surface_density_disks_kmt09`

**Description:** Implements the Krumholz et al. [2009] star formation rate surface density law for galactic disks.

**Code lines:** 208
19.1. Program units

**Contained by:** file `star_formation.rate_surface_density.disks.KMT09.F90`

**Modules used:**
- `galacticus_nodes`
- `kind_numbers`

**Used by:**
- subroutine `galacticus_calculations_reset`
- subroutine `star_formation_rate_surface_density_disks_initialize`

**function:** `kmt09_molecular_fraction_fast`

**Description:** Fast (but less accurate at low molecular fraction) fitting function from McKee and Krumholz [2010] for the molecular hydrogen fraction.

**Code lines:** 13

**Contained by:** module `star_formation_rate_surface_density_disks_kmt09`

**function:** `kmt09_molecular_fraction_slow`

**Description:** Slow (but more accurate at low molecular fraction) fitting function from Krumholz et al. [2009] for the molecular hydrogen fraction.

**Code lines:** 20

**Contained by:** module `star_formation_rate_surface_density_disks_kmt09`

**function:** `star_formation_rate_surface_density_disk_kmt09`

**Description:** Returns the star formation rate surface density (in $M_\odot$ Gyr$^{-1}$ Mpc$^{-2}$) for star formation in the galactic disk of thisNode. The disk is assumed to obey the Krumholz et al. [2009] star formation rule.

**Code lines:** 65

**Contained by:** module `star_formation_rate_surface_density_disks_kmt09`

**Modules used:**
- `abundances_structure`
- `galactic_structure_options`
- `galactic_structure_surface_densities`
- `numerical_constants_prefixes`

**subroutine:** `star_formation_rate_surface_density_disks_kmt09_initialize`

**Description:** Initializes the “KMT09” disk star formation rate surface density.

**Code lines:** 57

**Contained by:** module `star_formation_rate_surface_density_disks_kmt09`

**Modules used:**
- `input_parameters`
- `iso_varying_string`

**subroutine:** `star_formation_rate_surface_density_disks_kmt09_reset`

**Description:** Reset the extended Schmidt relation calculation.

**Code lines:** 8

**Contained by:** module `star_formation_rate_surface_density_disks_kmt09`

**file:** `star_formation.rate_surface_density.disks.Kennicutt-Schmidt.F90`

**Description:** Contains a module which implements the Kennicutt-Schmidt star formation rate surface density for galactic disks.

**Code lines:** 211

**module:** `star_formation_rate_surface_density_disks_ks`

**Description:** Implements the Kennicutt-Schmidt star formation rate surface density for galactic disks.

**Code lines:** 191

**Contained by:** file `star_formation.rate_surface_density.disks.Kennicutt-Schmidt.F90`

**Modules used:**
- `galacticus_nodes`
- `kind_numbers`

**Used by:**
- subroutine `galacticus_calculations_reset`
- subroutine `star_formation_rate_surface_density_disks_initialize`
function: star_formation_rate_surface_density_disk_ks

Description: Returns the star formation rate surface density (in $M_\odot$ Gyr$^{-1}$ Mpc$^{-2}$) for star formation in the galactic disk of thisNode. The disk is assumed to obey the Kennicutt-Schmidt law:

$$\Sigma_\star = A \left( \frac{x_h \Sigma_{\text{gas}}}{M_\odot \text{pc}^{-2}} \right)^N,$$

where $A = \text{[starFormationKennicuttSchmidtNormalization]}$ and $N = \text{[starFormationKennicuttSchmidtExponent]}$. Optionally, star formation is truncated for gas surface densities below a critical density of:

$$\Sigma_{\text{crit}} = \frac{q_{\text{crit}} \kappa \sigma_{\text{gas}}}{\pi G},$$

where $\kappa$ is the epicyclic frequency in the disk, $\sigma_{\text{gas}}$ is the velocity dispersion of gas in the disk and $q_{\text{crit}} = \text{[toomreParameterCritical]}$ is a dimensionless constant of order unity which controls where the critical density occurs. $\sigma_{\text{gas}}$ is assumed to be a constant equal to $\text{[velocityDispersionDiskGas]}$ and the disk is assumed to have a flat rotation curve such that $\kappa = \sqrt{2V/R}$.

Code lines: 60

Contained by: module star_formation_rate_surface_density_disks_ks

Modules used: abundances_structure, galactic_structure_options, galactic_structure_surface_densities, numerical_constants_math, numerical_constants_physical

subroutine: star_formation_rate_surface_density_disks_ks_initialize

Description: Initializes the “Kennicutt-Schmidt” disk star formation rate surface density.

Code lines: 88

Contained by: module star_formation_rate_surface_density_disks_ks

Modules used: input_parameters, iso_varying_string, numerical_constants_prefixes

subroutine: star_formation_rate_surface_density_disks_ks_reset

Description: Reset the Kennicutt-Schmidt relation calculation.

Code lines: 8

Contained by: module star_formation_rate_surface_density_disks_ks

file: star_formation_rate_surface_density_disks.extended_Schmidt.F90

Description: Contains a module which implements the extended Schmidt star formation rate surface density law of Shi et al. [2011] for galactic disks.

Code lines: 162

module: star_formation_rate_surface_density_disks_exschmidt

Description: Implements the extended Schmidt star formation rate surface density law of Shi et al. [2011] for galactic disks.

Code lines: 140

Contained by: file star_formation_rate_surface_density_disks.extended_Schmidt.F90

Modules used: galacticus_nodes, kind_numbers

Used by: subroutine galacticus_calculations_reset, subroutine star_formation_rate_surface_density_disks_ks_initialize
function: star_formation_rate_surface_density_disk_exschmidt
Description: Returns the star formation rate surface density (in $M_\odot$ Gyr$^{-1}$ Mpc$^{-2}$) for star formation in the galactic disk of thisNode. The disk is assumed to obey the extended Schmidt law of Shi et al. [2011]:

$$\dot{\Sigma}_* = A \left( \frac{x_H \Sigma_{\text{gas}}}{M_\odot \text{pc}^{-2}} \right)^{N_1} \left( \frac{\Sigma_*}{M_\odot \text{pc}^{-2}} \right)^{N_2},$$

(19.37)

where $A = \text{[starFormationExtendedSchmidtNormalization]}$ and $N_1 = \text{[starFormationExtendedSchmidtGasExponent]}$, $N_2 = \text{[starFormationExtendedSchmidtStarExponent]}$.

Code lines: 46
Contained by: module star_formation_rate_surface_density_disks_exschmidt
Modules used: abundances_structure galactic_structure_options galactic_structure_surface_densities

subroutine: star_formation_rate_surface_density_disks_exschmidt_initialize
Description: Initializes the “extended Schmidt” disk star formation rate surface density.
Code lines: 52
Contained by: module star_formation_rate_surface_density_disks_exschmidt
Modules used: input_parameters iso_varying_string numerical_constants_prefixes

subroutine: star_formation_rate_surface_density_disks_exschmidt_reset
Description: Reset the extended Schmidt relation calculation.
Code lines: 8
Contained by: module star_formation_rate_surface_density_disks_exschmidt


file: star_formation.timescales.disks.F90
Description: Contains a module which implements calculations of star formation timescales for galactic disks.
Code lines: 169

module: star_formation_timescales_disks
Description: Implements calculations of star formation timescales for galactic disks.
Code lines: 149
Contained by: file star_formation.timescales.disks.F90
Modules used: galacticus_nodes iso_c_binding iso_varying_string
Used by: function node_component_disk_-exponential_star_formation_rate function node_component_disk_very_-simple_sfr

function: star_formation_timescale_disk
Description: Returns the timescale (in Gyr) for star formation in the disk component of thisNode.
Code lines: 17
Contained by: module star_formation_timescales_disks
**subroutine**: star_formation_timescale_disks_initialize

*Description:* Initialize the disk star formation timescale module.

*Code lines:* 91

*Contained by:* module star_formation_timescales_disks

*Modules used:*
- galacticus_error
- input_parameters
- star_formation_timescale_disks_dynamical_time
- star_formation_timescale_disks_fixed
- star_formation_timescale_disks_halo_scaling
- star_formation_timescale_disks_integrated_sd

**file**: star_formation.timescales.disks.dynamical_time.F90

*Description:* Contains a module which implements a dynamical time-based star formation timescale for galactic disks.

*Code lines:* 130

**module**: star_formation_timescale_disks_dynamical_time

*Description:* Implements a dynamical time-based star formation timescale for galactic disks.

*Code lines:* 110

*Contained by:* file star_formation.timescales.disks.dynamical_time.F90

*Used by:* subroutine star_formation_timescale_disks_initialize

**function**: star_formation_timescale_disk_dynamical_time

*Description:* Returns the timescale (in Gyr) for star formation in the galactic disk of thisNode. The timescale is given by

\[ \tau \approx \epsilon^{-1} \tau_{\text{dynamical,disk}} \left( \frac{V_{\text{disk}}}{200 \text{km/s}} \right)^{\alpha}, \]  

where \( \epsilon \approx \text{starFormationDiskEfficiency} \) is a star formation efficiency and \( \alpha \approx \text{starFormationDiskVelocityExponent} \) controls the scaling with velocity. Note that \( \tau_{\text{dynamical,disk}} = \frac{R_{\text{disk}}}{V_{\text{disk}}} \) where the radius and velocity are whatever characteristic values returned by the disk method. This scaling is functionally similar to that adopted by Cole et al. [2000], but that they specifically used the half-mass radius and circular velocity at that radius.

*Code lines:* 38

*Contained by:* module star_formation_timescale_disks_dynamical_time

*Modules used:* galacticus_nodes numerical_constants_astronomical

**subroutine**: star_formation_timescale_disks_dynamical_time_initialize

*Description:* Initializes the “dynamical time” disk star formation timescale module.

*Code lines:* 54

*Contained by:* module star_formation_timescale_disks_dynamical_time

*Modules used:*
- array_utilities
- galacticus_error
- input_parameters
- iso_varying_string

**file**: star_formation.timescales.disks.fixed.F90

*Description:* Contains a module which implements a fixed star formation timescale for galactic disks.
19.1. Program units

**Code lines:** 72

**module:** star_formation_timescale_disks_fixed  
**Description:** Implements a fixed star formation timescale for galactic disks.  
**Code lines:** 52  
**Contained by:** file star_formation.timescales.disks.fixed.F90  
**Used by:** subroutine star_formation_timescale_disks_initialize

**function:** star_formation_timescale_disk_fixed  
**Description:** Returns the timescale (in Gyr) for star formation in the galactic disk of thisNode, assuming a fixed timecale.  
**Code lines:** 9  
**Contained by:** module star_formation_timescale_disks_fixed  
**Modules used:** galacticus_nodes

**subroutine:** star_formation_timescale_disks_fixed_initialize  
**Description:** Initializes the “fixed” disk star formation timescale module.  
**Code lines:** 25  
**Contained by:** module star_formation_timescale_disks_fixed  
**Modules used:** input_parameters iso_varying_string

**file:** star_formation.timescales.disks.halo_scaling.F90  
**Description:** Contains a module which implements a star formation timescale for galactic disks which scales with halo virial velocity and redshift.  
**Code lines:** 190

**module:** star_formation_timescale_disks_halo_scaling  
**Description:** Implements a star formation timescale for galactic disks which scales with halo virial velocity and redshift.  
**Code lines:** 152  
**Contained by:** file star_formation.timescales.disks.halo_scaling.F90  
**Modules used:** galacticus_nodes kind_numbers  
**Used by:** subroutine galacticus_calculations_reset subroutine star_formation_timescale_disks_initialize

**function:** star_formation_timescale_disk_halo_scaling  
**Description:** Returns the timescale (in Gyr) for star formation in the galactic disk of thisNode in the halo scaling timescale model.  
**Code lines:** 54  
**Contained by:** module star_formation_timescale_disks_halo_scaling  
**Modules used:** cosmology_functions dark_matter_halo_scales

**subroutine:** star_formation_timescale_disks_halo_scaling_initialize  
**Description:** Initializes the “halo scaling” disk star formation timescale module.  
**Code lines:** 52  
**Contained by:** module star_formation_timescale_disks_halo_scaling  
**Modules used:** input_parameters iso_varying_string
subroutine: star_formation_timescale_disks_halo_scaling_reset
   Description: Reset the halo scaling disk star formation timescale calculation.
   Code lines: 8
   Contained by: module star_formation_timescale_disks_halo_scaling

file: star_formation.timescales.disks.integrated_surface_density.F90
   Description: Contains a module which implements a global star formation timescale for galactic disks by integrating over the surface density of star formation rate.
   Code lines: 106

module: star_formation_timescale_disks_integrated_sd
   Description: Implements the global star formation timescale for galactic disks by integrating over the surface density of star formation rate.
   Code lines: 84
   Contained by: file star_formation.timescales.disks.integrated_surface_density.F90
   Modules used: galacticus_nodes
   Used by: subroutine star_formation_timescale_disks_initialize

function: star_formation_rate_integrand_surface_density
   Description: Integrand function for the “integrated surface density” star formation rate calculation.
   Code lines: 12
   Contained by: module star_formation_timescale_disks_integrated_sd
   Modules used: iso_c_binding star_formation_rate_surface_density_disks

function: star_formation_timescale_disk_integrated_sd
   Description: Returns the timescale (in Gyr) for star formation in the galactic disk of thisNode, by integrating over the surface density of star formation rate.
   Code lines: 41
   Contained by: module star_formation_timescale_disks_integrated_sd
   Modules used: iso_c_binding numerical_constants_math numerical_integration

subroutine: star_formation_timescale_disks_integrated_sd_initialize
   Description: Initializes the “integrated surface density” disk star formation timescale module.
   Code lines: 9
   Contained by: module star_formation_timescale_disks_integrated_sd
   Modules used: iso_varying_string

file: star_formation.timescales.spheroids.F90
   Description: Contains a module which implements calculations of star formation timescales for galactic spheroids.
   Code lines: 114

module: star_formation_timescales_spheroids
   Description: Implements calculations of star formation timescales for galactic spheroids.
   Code lines: 94
function: star_formation_timescale_spheroid
Description: Returns the timescale (in Gyr) for star formation in the spheroid component of thisNode.
Code lines: 12
Contained by: module star_formation_timescales_spheroids

subroutine: star_formation_timescale_spheroids_initialize
Description: Initialize the spheroid star formation timescale module.
Code lines: 53
Contained by: module star_formation_timescales_spheroids

file: star_formation.timescales.spheroids.dynamical_time.F90
Description: Contains a module which implements a dynamical time-based star formation timescale for galactic spheroids.
Code lines: 123

module: star_formation_timescale_spheroids_dynamical_time
Description: Implements a dynamical time-based star formation timescale for galactic spheroids.
Code lines: 103
Contained by: file star_formation.timescales.spheroids.dynamical_time.F90
Used by: subroutine star_formation_timescale_spheroids_initialize

function: star_formation_timescale_spheroids_dynamical_time
Description: Returns the timescale (in Gyr) for star formation in the galactic spheroid of thisNode. The timescale is given by
\[
\tau_* = \epsilon_*^{-1} \tau_{\text{dynamical,spheroid}} \left( \frac{V_{\text{spheroid}}}{200 \text{ km/s}} \right)^{\alpha_*},
\]
(19.39)
where \( \epsilon_* \) is a star formation efficiency and \( \alpha_* \) controls the scaling with velocity. Note that \( \tau_{\text{dynamical,spheroid}} = R_{\text{spheroid}}/V_{\text{spheroid}} \) where the radius and velocity are whatever characteristic values returned by the spheroid method. This scaling is functionally similar to that adopted by Cole et al. [2000], but that they specifically used the half-mass radius and circular velocity at that radius.

Code lines: 36
Contained by: module star_formation_timescale_spheroids_dynamical_time

subroutine: star_formation_timescale_spheroids_dynamical_time_initialize
Description: Initializes the “dynamical time” spheroid star formation timescale module.
19. Source Code Documentation

Code lines: 49
Contained by: module `star_formation_timescale_spheroids_dynamical_time`
Modules used: `input_parameters` `iso_varying_string`

File: `statistics.mass_function.covariance.F90`
Description: Contains a module which computes mass function covariances.
Code lines: 501

Module: `statistics_mass_function_covariance`
Description: Implements calculations of mass function covariances.
Code lines: 481
Contained by: file `statistics.mass_function.covariance.F90`
Modules used: `cosmology_functions`

Function: `bias_integrand_i`
Description: Integral for bias.
Code lines: 13
Contained by: module `statistics_mass_function_covariance`
Modules used: `dark_matter_halo_biases` `iso_c_binding`

Function: `halo_occupancy_integrand`
Description: Integral for mass function.
Code lines: 14
Contained by: module `statistics_mass_function_covariance`
Modules used: `conditional_mass_functions` `halo_mass_function` `iso_c_binding`

Function: `halo_occupancy_time_integrand`
Description: Integral for comoving volume.
Code lines: 21
Contained by: module `statistics_mass_function_covariance`
Modules used: `cosmology_functions` `fgsl` `iso_c_binding` `numerical_integration`

Function: `lss_integrand`
Description: Integral for LSS contribution to the covariance matrix.
Code lines: 35
Contained by: module `statistics_mass_function_covariance`
Modules used: `cosmology_functions` `fgsl` `iso_c_binding` `numerical_interpolation` `power_spectra_nonlinear`

Subroutine: `mass_function_covariance_matrix`
Description: Compute the mass function covariance matrix.
Code lines: 299
Contained by: module `statistics_mass_function_covariance`
Modules used: `fftw3` `fgsl` `galacticus_display` `galacticus_error`
function: mass_function_integrand_i
Description: Integral for mass function.
Code lines: 14
Contained by: module statistics_mass_function_covariance
Modules used: conditional_mass_functions halo_mass_function iso_c_binding

function: mass_function_time_integrand_i
Description: Integral for comoving volume.
Code lines: 21
Contained by: module statistics_mass_function_covariance
Modules used: cosmology_functions fgsl iso_c_binding numerical_integration

function: volume_integrand
Description: Integral for comoving volume.
Code lines: 11
Contained by: module statistics_mass_function_covariance
Modules used: cosmology_functions iso_c_binding

file: stellar_astrophysics.F90
Description: Contains a module which implements calculation of stellar astrophysics.
Code lines: 165

module: stellar_astrophysics
Description: Implements calculation of stellar astrophysics.
Code lines: 145
Contained by: file stellar_astrophysics.F90
Modules used: iso_varying_string
Used by: function metal_yield_integrand function recycled_fraction_integrand function star_is_evolved function stellar_feedback_cumulative_energy_input_standard function snepopiii_cumulative_energy_hegerwoosley function sneia_cumulative_number_nagashima

function: star_ejected_mass
Description: Returns the mass ejected by a star of given initialMass and metallicity.
Code lines: 12
Contained by: module stellar_astrophysics

function: star_initial_mass
Description: Returns the initial mass of a star of given lifetime and metallicity.
Code lines: 12
Contained by: module stellar_astrophysics
function: star_lifetime  
Description: Returns the lifetime of a star of given initialMass and metallicity.  
Code lines: 12  
Contained by: module stellar_astrophysics

function: star_metal_yield_mass  
Description: Returns the metal mass yielded by a star of given initialMass and metallicity.  
Code lines: 13  
Contained by: module stellar_astrophysics

subroutine: stellar_astrophysics_initialize  
Description: Initialize the stellar astrophysics module.  
Code lines: 52  
Contained by: module stellar_astrophysics

Modules used: galacticus_error  
modules input_parameters  
stellar_astrophysics_file

file: stellar_astrophysics.feedback.F90  
Description: Contains a module which provides calculations of stellar feedback.  
Code lines: 115

module: stellar_feedback  
Description: Provides calculations of stellar feedback.  
Code lines: 95  
Contained by: file stellar_astrophysics.feedback.F90

Modules used: iso_varying_string  
Used by: subroutine node_component_disk_very_-simple_rate_compute  
function cumulative_energy_integrand

subroutine star_formation_feedback_disk_-outflow_rate_creasey2012  
function star_formation_feedback_disk_-outflow_rate_fixed

subroutine star_formation_feedback_-disks_halo_scaling_initialize  
function star_formation_feedback_-outflow_rate_power_law

function star_formation_feedback_-spheroid_outflow_rate_power_law  
function star_formation_expulsive_-feedback_disk_outflow_rate_sw

function star_formation_expulsive_-feedback_spheroid_outflow_rate_sw  
subroutine stellar_population_-properties_rates_instantaneous

subroutine stellar_population_-properties_scales_noninstantaneous

function: stellar_feedback_cumulative_energy_input  
Description: Return the cumulative energy input per from stellar feedback from stars of given initialMass, age and metallicity.  
Code lines: 11  
Contained by: module stellar_feedback

subroutine: stellar_feedback_initialize  
Description: Initialize the stellar feedback module.  
Code lines: 52
19.1. Program units

Contained by:  module stellar_feedback
Modules used: galacticus_error input_parameters
stellar_feedback_standard

file: stellar_astrophysics.feedback.standard.F90
Description: Contains a module which implements a simple calculation of energy feedback from stellar populations.
Code lines: 140

module: stellar_feedback_standard
Description: Implements a simple calculation of energy feedback from stellar populations.
Code lines: 120
Contained by:  file stellar_astrophysics.feedback.standard.F90
Modules used: numerical_constants_astronomical
Used by:  subroutine stellar_feedback_initialize

function: stellar_feedback_cumulative_energy_input_standard
Description: Compute the cumulative energy input from a star of given initialMass, age and metallicity.
Code lines: 44
Contained by:  module stellar_feedback_standard
Modules used: iso_c_binding numerical_integration
stellar_astrophysics supernovae_population_iii
supernovae_type_ia

subroutine: stellar_feedback_standard_initialize
Description: Initialize the “standard” stellar feedback module.
Code lines: 39
Contained by:  module stellar_feedback_standard
Modules used: input_parameters iso_varying_string

function: wind_energy_integrand
Description: Integrand used in evaluating cumulative energy input from winds.
Code lines: 11
Contained by:  module stellar_feedback_standard
Modules used: iso_c_binding stellar_astrophysics_winds

file: stellar_astrophysics.file.F90
Description: Contains a module which implements calculation related to stellar astrophysics.
Code lines: 318

module: stellar_astrophysics_file
Description: Implements calculation related to stellar astrophysics.
Code lines: 298
Contained by:  file stellar_astrophysics.file.F90
Modules used: numerical_interpolation_2d_irregular
Used by:  subroutine stellar_astrophysics_file_initialize
function: star_ejected_mass_file
Description: Return the mass ejected during the lifetime of a star of given initialMass and metallicity.
Code lines: 13
Contained by: module stellar_astrophysics_file

function: star_initial_mass_file
Description: Return the initial mass of a star of given lifetime and metallicity.
Code lines: 11
Contained by: module stellar_astrophysics_file

function: star_lifetime_file
Description: Return the lifetime of a star (in Gyr) given an initialMass and metallicity.
Code lines: 12
Contained by: module stellar_astrophysics_file

function: star_metal_yield_mass_file
Description: Return the mass of metals yielded by a star of given initialMass and metallicity.
Code lines: 21
Contained by: module stellar_astrophysics_file

function: stellar_astrophysics_file_format_version
Description: Return the current file format version of stellar astrophysics files.
Code lines: 6
Contained by: module stellar_astrophysics_file

subroutine: stellar_astrophysics_file_initialize
Description: Initialize the stellar astrophysics module.
Code lines: 196
Contained by: module stellar_astrophysics_file
Modules used: atomic_data
                fox_dom
                galacticus_error
                galacticus_input_paths
                input_parameters
                io_xml
                iso_varying_string
                memory_management

file: stellar_astrophysics.supernovae_PopulationIII.F90
Description: Contains a module which provides calculations of Population III supernovae.
Code lines: 110

module: supernovae_population_iii
Description: Provides calculations of Population III supernovae.
Code lines: 90
Contained by: file stellar_astrophysics.supernovae_PopulationIII.F90
Modules used: iso_varying_string
Used by: function stellar_feedback_cumulative_energy

function: snepopiii_cumulative_energy
Description: Return the cumulative energy input from Population III supernovae from stars of given initialMass, age and metallicity.
19.1. Program units

Subroutine: supernovae_population_iii_increase
Description: Initialize the Population III supernovae module.
Code lines: 52

Modules used:
- galacticus_error
- input_parameters
- supernovae_population_iii_hegerwoosley

Description: Contains a module which implements calculations related to Population III supernovae.
Code lines: 112

Module: supernovae_population_iii_hegerwoosley
Description: Implements calculations related to Population III supernovae.
Code lines: 92


Used by:
- subroutine supernovae_population_iii_increase

Function: supernovae_cumulative_energy_hegerwoosley
Description: Compute the cumulative energy input from Population III star pair instability supernovae using the results of Heger and Woosley [2002].
Code lines: 32

Modules used:
- fgs1
- numerical_interpolation
- stellar_astrophysics

Subroutine: supernovae_population_iii_hegerwoosley_increase
Description: Initialize the “Heger-Woosley2002” Population III supernovae module.
Code lines: 41

Modules used:
- fox_dom
- galacticus_error
- io_xml
- iso_varying_string
- numerical_constants_astronomical

File: stellar_astrophysics.supernovae.type.Ia.F90
Description: Contains a module which provides calculations of Type Ia supernovae.
Code lines: 131

Module: supernovae_type_ia
Description: Provides calculations of Type Ia supernovae.
Code lines: 111

Contained by: file stellar_astrophysics.supernovae.type.Ia.F90

Used by:
- function metal_yield_integrand
- function stellar_feedback_cumulative_energy_input_standard
19. Source Code Documentation

**function: sneia_cumulative_number**
*Description:* Return the cumulative number of Type Ia supernovae from stars of given initialMass, age and metallicity.
*Code lines:* 11
*Contained by:* module supernovae_type_ia

**function: sneia_cumulative_yield**
*Description:* Return the cumulative yield of Type Ia supernovae from stars of given initialMass, age and metallicity.
*Code lines:* 12
*Contained by:* module supernovae_type_ia

**subroutine: supernovae_type_ia_initialize**
*Description:* Initialize the Type Ia supernovae module.
*Code lines:* 52
*Contained by:* module supernovae_type_ia

**file: stellar_astrophysics.supernovae_type_Ia.Nagashima.F90**
*Description:* Contains a module which implements calculations related to Type Ia supernovae.
*Code lines:* 149

**module: supernovae_type_ia_nagashima**
*Description:* Implements calculations related to Type Ia supernovae.
*Code lines:* 129
*Contained by:* file stellar_astrophysics.supernovae_type_Ia.Nagashima.F90
*Used by:* subroutine supernovae_type_ia_initialize

**function: sneia_cumulative_number_nagashima**
*Description:* Compute the cumulative number of Type Ia supernovae originating per unit mass of stars that form with given initialMass and metallicity after a time age. The calculation is based on that of Nagashima et al. [2005]. The number returned here assumes a distribution of binary mass ratios and so only makes sense once it is integrated over an initial mass function.
*Code lines:* 29
*Contained by:* module supernovae_type_ia_nagashima
*Modules used:* stellar_astrophysics

**function: sneia_cumulative_yield_nagashima**
*Description:* Compute the cumulative yield from Type Ia supernovae originating per unit mass of stars that form with given initialMass and metallicity after a time age. The calculation is based on the Type Ia rate calculation of Nagashima et al. [2005] and the Type Ia yields from Nomoto et al. [1997]. The number returned here assumes a distribution of binary mass ratios and so only makes sense once it is integrated over an initial mass function.
*Code lines:* 19
*Contained by:* module supernovae_type_ia_nagashima

**subroutine: supernovae_type_ia_nagashima_initialize**
19.1. Program units

**Description:** Initialize the “Nagashima” Type Ia supernovae module.

**Code lines:** 56

**Contained by:** module `supernovae_type_ia_nagashima`

**Modules used:**
- `atomic_data`
- `fox_dom`
- `galacticus_error`
- `galacticus_input_paths`
- `io_xml`
- `iso_varying_string`
- `memory_management`

**file:** `stellar_astrophysics.tracks.F90`

**Description:** Contains a module which implements calculation of stellar tracks.

**Code lines:** 127

**module:** `stellar_astrophysics_tracks`

**Description:** Implements stellar tracks.

**Code lines:** 107

**Contained by:** file `stellar_astrophysics.tracks.F90`

**Modules used:** `iso_varying_string`

**Used by:**
- function `stellar_winds_mass_loss_rate_leitherer1992`
- function `stellar_winds_terminal_velocity_leitherer1992`

**function:** `stellar_effective_temperature`

**Description:** Returns the effective temperature of a star of given `initialMass`, `metallicity` and `age`.

**Code lines:** 12

**Contained by:** module `stellar_astrophysics_tracks`

**function:** `stellar_luminosity`

**Description:** Returns the bolometric luminosity of a star of given `initialMass`, `metallicity` and `age`.

**Code lines:** 12

**Contained by:** module `stellar_astrophysics_tracks`

**subroutine:** `stellar_tracks_initialize`

**Description:** Initialize the cosmology functions module.

**Code lines:** 52

**Contained by:** module `stellar_astrophysics_tracks`

**Modules used:**
- `galacticus_error`
- `input_parameters`
- `stellar_astrophysics_tracks_file`

**file:** `stellar_astrophysics.tracks.file.F90`

**Description:** Contains a module which implements calculation of stellar tracks.

**Code lines:** 338

**module:** `stellar_astrophysics_tracks_file`

**Description:** Implements stellar tracks.

**Code lines:** 318

**Contained by:** file `stellar_astrophysics.tracks.file.F90`

**Modules used:** `fgsl`

**Used by:** subroutine `stellar_tracks_initialize`
function: stellar_effective_temperature_file
Description: Return the effective temperature (in Kelvin) for a star of given initialMass, metallicity and age.
Code lines: 20
Contained by: module stellar_astrophysics_tracks_file
Modules used: iso_c_binding

function: stellar_luminosity_file
Description: Return the bolometric luminosity (in $L_\odot$) for a star of given initialMass, metallicity and age.
Code lines: 20
Contained by: module stellar_astrophysics_tracks_file
Modules used: iso_c_binding

subroutine: stellar_tracks_initialize_file
Description: Initialize the stellar tracks module.
Code lines: 133
Contained by: module stellar_astrophysics_tracks_file
Modules used: galacticus_error, galacticus_input_paths, input_parameters, io_hdf5, iso_varying_string, memory_management, string_handling

function: stellar_tracks_interpolation_do
Description: Using precomputed factors, interpolate in metallicity, mass and age in the given stellarTracks.
Code lines: 22
Contained by: module stellar_astrophysics_tracks_file
Modules used: iso_c_binding

subroutine: stellar_tracks_interpolation_get
Description: Get interpolating factors for stellar tracks.
Code lines: 89
Contained by: module stellar_astrophysics_tracks_file
Modules used: iso_c_binding, numerical_interpolation

file: stellar_astrophysics.winds.F90
Description: Contains a module which provides calculations of stellar winds.
Code lines: 124

module: stellar_astrophysics_winds
Description: Provides calculations of stellar winds.
Code lines: 104
Contained by: file stellar_astrophysics.winds.F90
Modules used: iso_varying_string
Used by: function wind_energy_integrand
subroutine: stellar_winds_initialize
19.1. Program units

**Description:** Initialize the stellar winds module.

**Code lines:** 52

**Contained by:** module `stellar_astrophysics_winds`

**Modules used:**
- `galacticus_error`
- `input_parameters`
- `stellar_astrophysics_winds_leitherer1992`

**Function:** `stellar_winds_mass_loss_rate`

**Description:** Return the mass loss rate (in $M_\odot$/Gyr) from stars of given initialMass, age and metallicity.

**Code lines:** 11

**Contained by:** module `stellar_astrophysics_winds`

**Function:** `stellar_winds_terminal_velocity`

**Description:** Return the terminal velocity (in km/s) of winds from stars of given initialMass, age and metallicity.

**Code lines:** 11

**Contained by:** module `stellar_astrophysics_winds`

**File:** `stellar_astrophysics.winds.Leitherer1992.F90`

**Description:** Contains a module which implements a calculation of winds from stellar populations using the fitting formulae of Leitherer et al. [1992].

**Code lines:** 93

**Module:** `stellar_astrophysics_winds_leitherer1992`

**Description:** Implements a calculation of winds from stellar populations using the fitting formulae of Leitherer et al. [1992].

**Code lines:** 73

**Contained by:** file `stellar_astrophysics.winds.Leitherer1992.F90`

**Modules used:** `numerical_constants_astronomical`

**Used by:** subroutine `stellar_winds_initialize`

**Subroutine:** `stellar_winds_leitherer1992_initialize`

**Description:** Initialize the "Leitherer1992" stellar winds module.

**Code lines:** 14

**Contained by:** module `stellar_astrophysics_winds_leitherer1992`

**Modules used:** `iso_varying_string`

**Function:** `stellar_winds_mass_loss_rate_leitherer1992`

**Description:** Compute the mass loss rate (in $M_\odot$/Gyr) from a star of given initialMass, age and metallicity using the fitting formula of Leitherer et al. [1992].

**Code lines:** 19

**Contained by:** module `stellar_astrophysics_winds_leitherer1992`

**Modules used:** `stellar_astrophysics_tracks`

**Function:** `stellar_winds_terminal_velocity_leitherer1992`

**Description:** Compute the terminal velocity (in km/s) from a star of given initialMass, age and metallicity using the fitting formula of Leitherer et al. [1992].

**Code lines:** 19

**Contained by:** module `stellar_astrophysics_winds_leitherer1992`
Modules used: stellar_astrophysics_tracks

file: stellar_populations.luminosities.F90
Description: Contains a module which implements calculations of stellar population luminosities in the AB magnitude system.
Code lines: 361

module: stellar_population_luminosities
Description: Implements calculations of stellar population luminosities in the AB magnitude system.
Code lines: 341
Contained by: file stellar_populations.luminosities.F90
Modules used: abundances_structure fgs1 iso_c_binding
Used by: subroutine stellar_luminosities_set

function: filter_luminosity_integrand
Description: Integrand for the luminosity through a given filter.
Code lines: 21
Contained by: module stellar_population_luminosities
Modules used: instruments_filters stellar_population_spectra

function: filter_luminosity_integrand_ab
Description: Integrand for the luminosity of a zeroth magnitude (AB) source through a given filter.
Code lines: 13
Contained by: module stellar_population_luminosities
Modules used: instruments_filters numerical_constants_astronomical

type: luminositytable
Description: Structure for holding tables of simple stellar population luminosities.
Code lines: 9
Contained by: module stellar_population_luminosities

function: stellar_population_luminosity
Description: Returns the luminosity for a $1M_\odot$ simple stellar population of given abundances and age drawn from IMF specified by imfIndex and observed through the filter specified by filterIndex.
Code lines: 261
Contained by: module stellar_population_luminosities
Modules used: file_utilities galacticus_display galacticus_input_paths
input_parameters instruments_filters
io_hdf5 iso_varying_string
memory_management numerical_constants_astronomical
numerical_integration numerical_interpolation
star_formation_imf string_handling
stellar_population_spectra_-
postprocess
file: stellar_populations.properties.F90
Description: Contains a module which provides support for stellar population properties.
Code lines: 166

module: stellar_population_properties
Description: Provides support for stellar population properties.
Code lines: 146
Contained by: file stellar_populations.properties.F90
Modules used: abundances_structure
histories
exponential_create
exponential_scale_set
modules used galacticus_nodes iso_varying_string

Module: module stellar_population_properties
Description: Provides support for stellar population properties.
Code lines: 146
Contained by: file stellar_populations.properties.F90
Modules used: abundances_structure
histories
exponential_create
exponential_scale_set
modules used galacticus_nodes iso_varying_string

module: module stellar_population_properties
Description: Provides support for stellar population properties.
Code lines: 146
Contained by: file stellar_populations.properties.F90
Modules used: abundances_structure
histories
exponential_create
exponential_scale_set
modules used galacticus_nodes iso_varying_string

Used by:
subroutine node_component_disk_exponential_create
subroutine node_component_disk_exponential_rate_compute
subroutine node_component_disk_exponential_scale_set
module node_component_spheroid_standard

function: function stellar_population_properties_history_count
Description: Return a count of the number of histories which must be stored for the selected stellar populations method.
Code lines: 11
Contained by: module stellar_population_properties

subroutine: subroutine stellar_population_properties_history_create
Description: Create any history required for storing stellar population properties.
Code lines: 13
Contained by: module stellar_population_properties

subroutine: subroutine stellar_population_properties_rates
Description: Return an array of stellar population property rates of change given a star formation rate and fuel abundances.
Code lines: 19
Contained by: module stellar_population_properties
Modules used: stellar_luminosities_structure

subroutine: subroutine stellar_population_properties_rates_initialize
Description: Initialize the disk star formation timecale module.
Code lines: 53
Contained by: module stellar_population_properties
Modules used: galacticus_error
input_parameters
stellar_population_properties_instantaneous
stellar_population_properties_noninstantaneous

subroutine: subroutine stellar_population_properties_scales
Description: Set the scaling factors for error control on the absolute value of stellar population properties.
Code lines: 13
Contained by: module stellar_population_properties

file: stellar_populations.properties.instantaneous.F90
Description: Contains a module which implements stellar population properties in the instantaneous recycling approximation.
module: stellar_population_properties_instantaneous
Description: Implements stellar population properties in the instantaneous recycling approximation.
Code lines: 127
Contained by: file stellar_populations.properties.instantaneous.F90
Used by: subroutine stellar_population_properties_rates_initialize

function: stellar_population_properties_history_count_instantaneous
Description: Returns the number of histories required by the instantaneous stellar populations properties module.
Code lines: 7
Contained by: module stellar_population_properties_instantaneous

subroutine: stellar_population_properties_history_create_instantaneous
Description: Create any history required for storing stellar population properties. The instantaneous method requires none, so don’t create one.
Code lines: 10
Contained by: module stellar_population_properties_instantaneous
Modules used: galacticus_nodes histories

subroutine: stellar_population_properties_instantaneous_initialize
Description: Initializes the instantaneous recycling approximation stellar population properties module.
Code lines: 21
Contained by: module stellar_population_properties_instantaneous
Modules used: atomic_data iso_varying_string

subroutine: stellar_population_properties_rates_instantaneous
Description: Return an array of stellar population property rates of change given a star formation rate and fuel abundances.
Code lines: 53
Contained by: module stellar_population_properties_instantaneous
Modules used: abundances_structure galacticus_nodes histories star_formation_imf stellar_feedback stellar_luminosities_structure

subroutine: stellar_population_properties_scales_instantaneous
Description: Set the scalings for error control on the absolute values of stellar population properties. The instantaneous method requires none, so just return.
Code lines: 12
Contained by: module stellar_population_properties_instantaneous
Modules used: abundances_structure histories

file: stellar_populations.properties.noninstantaneous.F90
Description: Contains a module which implements stellar population properties with noninstantaneous recycling.
Code lines: 250

module: stellar_population_properties_noninstantaneous
19.1. Program units

Description: Implements stellar population properties with noninstantaneous recycling.
Code lines: 230
Contained by: file stellar_populations.properties.noninstantaneous.F90
Used by: subroutine stellar_population_properties_rates_initialize

function: stellar_population_properties_history_count_noninstantaneous
Description: Returns the number of histories required by the noninstantaneous stellar populations properties module.
Code lines: 7
Contained by: module stellar_population_properties_noninstantaneous

subroutine: stellar_population_properties_history_create_noninstantaneous
Description: Create any history required for storing stellar population properties.
Code lines: 20
Contained by: module stellar_population_properties_noninstantaneous
Modules used: galacticus_nodes histories numerical_ranges

subroutine: stellar_population_properties_noninstantaneous_initialize
Description: Initializes the noninstantaneous recycling stellar population properties module.
Code lines: 42
Contained by: module stellar_population_properties_noninstantaneous
Modules used: abundances_structure input_parameters iso_varying_string

subroutine: stellar_population_properties_rates_noninstantaneous
Description: Return an array of stellar population property rates of change given a star formation rate and fuel abundances.
Code lines: 85
Contained by: module stellar_population_properties_noninstantaneous
Modules used: abundances_structure fgsl galacticus_nodes histories iso_c_binding numerical_interpolation star_formation_imf stellar_luminosities_structure

subroutine: stellar_population_properties_scales_noninstantaneous
Description: Set the scalings for error control on the absolute values of stellar population properties.
Code lines: 42
Contained by: module stellar_population_properties_noninstantaneous
Modules used: abundances_structure histories memory_management stellar_feedback

file: stellar_populations.spectra.Conroy_et_al.F90
Description: Contains a module which handles stellar spectra using the Conroy et al. [2009] package.
Code lines: 164

module: stellar_population_spectra_conroy
Description: Handles stellar spectra using the Conroy et al. [2009] package.
19. Source Code Documentation

Code lines: 144
Contained by: file stellar_populations.spectra.Conroy_et_al.F90
Modules used: iso_varying_string
Used by: module stellar_population_spectra

function: stellar_population_spectra_conroy_get
Description: Return the luminosity (in units of $L_\odot \, Hz^{-1}$) for a stellar population with composition abundances, of the given age (in Gyr) and the specified wavelength (in Angstroms). This is computed using the Conroy et al. [2009] package.
Code lines: 19
Contained by: module stellar_population_spectra_conroy
Modules used: abundances_structure stellar_population_spectra_conroy stellar_population_spectra_file

subroutine: stellar_population_spectra_conroy_initialize
Description: Initializes the “Conroy-White-Gunn2009” module.
Code lines: 12
Contained by: module stellar_population_spectra_conroy

subroutine: stellar_population_spectra_conroy_initialize_imf
Description: Ensure that the requested IMF has been generated and loaded.
Code lines: 75
Contained by: module stellar_population_spectra_conroy
Modules used: file_utilities galacticus_input_paths io_hdf5 memory_management star_formation_imf stellar_population_spectra_file string_handling system_command tables

subroutine: stellar_population_spectrum_tabulation_conroy
Description: Return a tabulation of ages and metallicities at which stellar spectra for the specified IMF should be tabulated.
Code lines: 15
Contained by: module stellar_population_spectra_conroy
Modules used: stellar_population_spectra_file

file: stellar_populations.spectra.F90
Description: Contains a module that implements calculations of stellar population spectra.
Code lines: 145

module: stellar_population_spectra
Description: Implements calculations of stellar population spectra.
Code lines: 125
Contained by: file stellar_populations.spectra.F90
Modules used: abundances_structure iso_varying_string
stellar_population_spectra_conroy stellar_population_spectra_file
Used by: function intergalactic_medium_state_-_internal_update
function radiation_intergalactic_-_background_internal_update
function stellarspectraconvolution
function filter_luminosity_integrand
function `stellar_population_luminosity`

**function:** `stellar_population_spectrum`

*Description:* Return the luminosity (in units of \( L_\odot \) Hz\(^{-1}\)) for a stellar population with composition abundances, of the given age (in Gyr) and the specified wavelength (in Angstroms).

*Code lines:* 16
*Contained by:* module `stellar_population_spectra`

**subroutine:** `stellar_population_spectrum_initialize`

*Description:* Initialize the stellar population spectra module

*Code lines:* 42
*Contained by:* module `stellar_population_spectra`
*Modules used:* `galacticus_error` `input_parameters`

**subroutine:** `stellar_population_spectrum_tabulation`

*Description:* Return a tabulation of ages and metallicities at which stellar spectra for the specified IMF should be tabulated.

*Code lines:* 13
*Contained by:* module `stellar_population_spectra`

**file:** `stellar_populations.spectra.file.F90`

*Description:* Contains a module which reads and interpolates a file of stellar population spectra.

*Code lines:* 367

**module:** `stellar_population_spectra_file`

*Description:* Reads and interpolates a file of stellar population spectra.

*Code lines:* 347
*Contained by:* file `stellar_populations.spectra.file.F90`
*Modules used:* `fgsl` `iso_varying_string`
*Used by:* function `stellar_population_spectra_conroy_get` subroutine `stellar_population_spectra_conroy_initialize_imf` subroutine `stellar_population_spectra_spectrum_tabulation_conroy`

**type:** `spectraltable`

*Description:* Structure to hold spectral data.

*Code lines:* 10
*Contained by:* module `stellar_population_spectra_file`

**function:** `stellar_population_spectra_file_format_current`

*Description:* Return the current file format version for stellar spectra files.

*Code lines:* 6
*Contained by:* module `stellar_population_spectra_file`

**function:** `stellar_population_spectra_file_get`

*Description:* Return the luminosity (in units of \( L_\odot \) Hz\(^{-1}\)) for a stellar population with composition abundances, of the given age (in Gyr) and the specified wavelength (in Angstroms). This is found by interpolating in tabulated spectra.

*Code lines:* 18
*Contained by:* module `stellar_population_spectra_file`
*Modules used:* `abundances_structure`
**subroutine:** stellar_population_spectra_file_initialize

*Description:* Initializes the “stellar population spectra from file” module.

*Code lines:* 12

*Contained by:* module stellar_population_spectra_file

**subroutine:** stellar_population_spectra_file_initialize_imf

*Description:* Ensure that data is loaded for the requested IMF.

*Code lines:* 71

*Contained by:* module stellar_population_spectra_file

*Modules used:* galacticus_input_paths input_parameters star_formation_imf

**function:** stellar_population_spectra_file_interpolate

*Description:* Compute the stellar spectrum by interpolation in the tabulated data.

*Code lines:* 85

*Contained by:* module stellar_population_spectra_file

*Modules used:* abundances_structure galacticus_error iso_c_binding numerical_interpolation

**subroutine:** stellar_population_spectra_file_read

*Description:* Read a file of simple stellar population spectra.

*Code lines:* 78

*Contained by:* module stellar_population_spectra_file

*Modules used:* galacticus_error io_hdf5 memory_management

**subroutine:** stellar_population_spectra_file_tabulation

*Description:* Return a tabulation of ages and metallicities at which stellar spectra for the specified IMF should be tabulated.

*Code lines:* 23

*Contained by:* module stellar_population_spectra_file

*Modules used:* memory_management numerical_constants_astronomical

**file:** stellar_populations_spectra_postprocess.F90

*Description:* Contains a module which provides an object that implements stellar spectra postprocessors.

*Code lines:* 954

**module:** stellar_population_spectra_postprocess
19.1. Program units

**Description:** Provides an object that implements stellar spectra postprocessors. An implementation of a spectrum postprocessor that does nothing. An implementation of a spectrum postprocessor that keeps only recent populations. An implementation of a spectrum postprocessor that suppresses the Lyman continuum. An implementation of a spectrum postprocessor that applies the Inoue et al. [2014] calculation of the attenuation of spectra by the intergalactic medium. An implementation of a spectrum postprocessor that applies the Madau [1995] calculation of the attenuation of spectra by the intergalactic medium. An implementation of a spectrum postprocessor that applies the Meiksin [2006] calculation of the attenuation of spectra by the intergalactic medium.

**Code lines:** 934

**Contained by:** file `stellar_populations.spectra.postprocess.F90`

**Modules used:** `iso_c_binding` `iso_varying_string`

**Used by:** subroutine `stellar_luminosities_initialize`
function `filter_luminosity_integrand`
function `stellar_population_luminosity`
program `test_inoue2014`

**subroutine:** `identityapply`
19. Source Code Documentation

Description: Perform an identity postprocessing on a spectrum.

Code lines: 9

Contained by: module stellar_population_spectra_postprocess

function: identitydefaultconstructor

Description: Default constructor for the identity spectrum postprocessor class.

Code lines: 6

Contained by: module stellar_population_spectra_postprocess

subroutine: inoue2014apply

Description: Suppress the Lyman continuum in a spectrum.

Code lines: 73

Contained by: module stellar_population_spectra_postprocess

Modules used: factorials gamma_functions numerical_constants_atomic

function: inoue2014defaultconstructor

Description: Default constructor for the inoue2014 spectrum postprocessor class.

Code lines: 6

Contained by: module stellar_population_spectra_postprocess

subroutine: lycsuppressapply

Description: Suppress the Lyman continuum in a spectrum.

Code lines: 10

Contained by: module stellar_population_spectra_postprocess

Modules used: numerical_constants_atomic

function: lycsuppressdefaultconstructor

Description: Default constructor for the lycSuppress spectrum postprocessor class.

Code lines: 6

Contained by: module stellar_population_spectra_postprocess

subroutine: madau1995apply

Description: Suppress the Lyman continuum in a spectrum.

Code lines: 38

Contained by: module stellar_population_spectra_postprocess

Modules used: numerical_constants_atomic

function: madau1995defaultconstructor

Description: Default constructor for the madau1995 spectrum postprocessor class.

Code lines: 6

Contained by: module stellar_population_spectra_postprocess

subroutine: meiksin2006apply

Description: Suppress the Lyman continuum in a spectrum.

Code lines: 85

Contained by: module stellar_population_spectra_postprocess

Modules used: factorials gamma_functions numerical_constants_atomic

1836
function: meiksin2006defaultconstructor
Description: Default constructor for the meiksin2006 spectrum postprocessor class.
Code lines: 6
Contained by: module stellar_population_spectra_postprocess

type: postprocessor
Code lines: 2
Contained by: module stellar_population_spectra_postprocess

type: postprocessors
Code lines: 3
Contained by: module stellar_population_spectra_postprocess

subroutine: recentapply
Description: Perform an recent postprocessing on a spectrum.
Code lines: 9
Contained by: module stellar_population_spectra_postprocess

function: recentdefaultconstructor
Description: Default constructor for the recent spectrum postprocessor class.
Code lines: 23
Contained by: module stellar_population_spectra_postprocess

Modules used: input_parameters

function: recentgenericconstructor
Description: Generic constructor for the recent spectrum postprocessor class.
Code lines: 8
Contained by: module stellar_population_spectra_postprocess

interface: spectrapostprocessor
Code lines: 3
Contained by: module stellar_population_spectra_postprocess

subroutine: spectrapostprocessorapplynull
Description: Apply postprocessing to a spectrum.
Code lines: 9
Contained by: module stellar_population_spectra_postprocess
Modules used: galacticus_error

type: spectrapostprocessorclass
Code lines: 21
Contained by: module stellar_population_spectra_postprocess

function: spectrapostprocessorconstructordefault
Description: Return a pointer to the default spectrapostprocessor object.
Code lines: 8
Contained by: module stellar_population_spectra_postprocess
function: spectrapostprocessorconstructornamed
Description: Return a pointer to a newly created spectraPostprocessor object of the specified type.
Code lines: 58
Contained by: module stellar_population_spectra_postprocess
Modules used: galacticus_error iso_varying_string

interface: spectrapostprocessoridentity
Description: Constructors for the identity spectrum postprocessor class.
Code lines: 3
Contained by: module stellar_population_spectra_postprocess

subroutine: spectrapostprocessorinitialize
Description: Initialize the default spectraPostprocessor object.
Code lines: 75
Contained by: module stellar_population_spectra_postprocess
Modules used: galacticus_error input_parameters iso_varying_string

interface: spectrapostprocessorinoue2014
Description: Constructors for the inoue2014 spectrum postprocessor class.
Code lines: 3
Contained by: module stellar_population_spectra_postprocess

function: spectrapostprocessorisfinalizable
Description: Return true if this object can be finalized.
Code lines: 7
Contained by: module stellar_population_spectra_postprocess

interface: spectrapostprocessorlycsuppress
Description: Constructors for the lycSuppress spectrum postprocessor class.
Code lines: 3
Contained by: module stellar_population_spectra_postprocess

interface: spectrapostprocessormadau1995
Description: Constructors for the madau1995 spectrum postprocessor class.
Code lines: 3
Contained by: module stellar_population_spectra_postprocess

interface: spectrapostprocessormeiksin2006
Description: Constructors for the meiksin2006 spectrum postprocessor class.
Code lines: 3
Contained by: module stellar_population_spectra_postprocess

interface: spectrapostprocessorrecent
Description: Constructors for the recent spectrum postprocessor class.
Code lines: 4
Contained by: module stellar_population_spectra_postprocess
function: stellar_population_spectrum_postprocess
Description: Return a multiplicative factor by which a stellar population spectrum should be modified by any postprocessing.
Code lines: 16
Contained by: module stellar_population_spectra_postprocess

function: stellar_population_spectrum_postprocess_chain_methods
Description: Return a label describing the postprocessing methods used in the indicated chain.
Code lines: 8
Contained by: module stellar_population_spectra_postprocess

function: stellar_population_spectrum_postprocess_index
Description: Return the index to the specified postprocessing chain.
Code lines: 90
Contained by: module stellar_population_spectra_postprocess
Modules used: galacticus_error input_parameters string_handling

subroutine: stellar_population_spectrum_postprocess_initialize
Description: Initialize the stellar population spectra postprocessing module
Code lines: 18
Contained by: module stellar_population_spectra_postprocess

file: stellar_populations.spectra.postprocess.Inoue2014.F90
Description: An implementation of a spectrum postprocessor that applies the Inoue et al. [2014] calculation of the attenuation of spectra by the intergalactic medium.
Code lines: 124

subroutine: inoue2014apply
Description: Suppress the Lyman continuum in a spectrum.
Code lines: 73
Contained by: file stellar_populations.spectra.postprocess.Inoue2014.F90
Modules used: factorials gamma_functions numerical_constants_atomic

function: inoue2014defaultconstructor
Description: Default constructor for the inoue2014 spectrum postprocessor class.
Code lines: 6
Contained by: file stellar_populations.spectra.postprocess.Inoue2014.F90

interface: spectrapostprocessorinoue2014
Description: Constructors for the inoue2014 spectrum postprocessor class.
Code lines: 3
Contained by: file stellar_populations.spectra.postprocess.Inoue2014.F90

file: stellar_populations.spectra.postprocess.Lyman_continuum_suppress.F90
Description: An implementation of a spectrum postprocessor that suppresses the Lyman continuum.
Code lines: 56
19. Source Code Documentation

subroutine: lycsuppressapply
Description: Suppress the Lyman continuum in a spectrum.
Code lines: 10
Contained by: file stellar_populations.spectra.postprocess.Lyman_continuum_suppress.F90
Modules used: numerical_constants_atomic

function: lycsuppressdefaultconstructor
Description: Default constructor for the lycSuppress spectrum postprocessor class.
Code lines: 6
Contained by: file stellar_populations.spectra.postprocess.Lyman_continuum_suppress.F90

interface: spectrapostprocessorlycsuppress
Description: Constructors for the lycSuppress spectrum postprocessor class.
Code lines: 3
Contained by: file stellar_populations.spectra.postprocess.Lyman_continuum_suppress.F90

Description: An implementation of a spectrum postprocessor that applies the Madau [1995] calculation of the attenuation of spectra by the intergalactic medium.
Code lines: 85

subroutine: madau1995apply
Description: Suppress the Lyman continuum in a spectrum.
Code lines: 38
Modules used: numerical_constants_atomic

function: madau1995defaultconstructor
Description: Default constructor for the madau1995 spectrum postprocessor class.
Code lines: 6

interface: spectrapostprocessormadau1995
Description: Constructors for the madau1995 spectrum postprocessor class.
Code lines: 3

file: stellar_populations.spectra.postprocess.Meiksin2006.F90
Description: An implementation of a spectrum postprocessor that applies the Meiksin [2006] calculation of the attenuation of spectra by the intergalactic medium.
Code lines: 132

subroutine: meiksin2006apply
Description: Suppress the Lyman continuum in a spectrum.
Code lines: 85
Contained by: file stellar_populations.spectra.postprocess.Meiksin2006.F90
Modules used: factorials gamma_functions numerical_constants_atomic
function: meiksin2006defaultconstructor
Description: Default constructor for the meiksin2006 spectrum postprocessor class.
Code lines: 6
Contained by: file stellar_populations.spectra.postprocess.Meiksin2006.F90

interface: spectrapostprocessorsmeiksin2006
Description: Constructors for the meiksin2006 spectrum postprocessor class.
Code lines: 3
Contained by: file stellar_populations.spectra.postprocess.Meiksin2006.F90

file: stellar_populations.spectra.postprocess.identity.F90
Description: An implementation of a spectrum postprocessor that does nothing.
Code lines: 55

subroutine: identityapply
Description: Perform an identity postprocessing on a spectrum.
Code lines: 9
Contained by: file stellar_populations.spectra.postprocess.identity.F90

function: identitydefaultconstructor
Description: Default constructor for the identity spectrum postprocessor class.
Code lines: 6
Contained by: file stellar_populations.spectra.postprocess.identity.F90

interface: spectrapostprocessorentity
Description: Constructors for the identity spectrum postprocessor class.
Code lines: 3
Contained by: file stellar_populations.spectra.postprocess.identity.F90

file: stellar_populations.spectra.postprocess.recent.F90
Description: An implementation of a spectrum postprocessor that keeps only recent populations.
Code lines: 87

subroutine: recentapply
Description: Perform an recent postprocessing on a spectrum.
Code lines: 9
Contained by: file stellar_populations.spectra.postprocess.recent.F90

function: recentdefaultconstructor
Description: Default constructor for the recent spectrum postprocessor class.
Code lines: 23
Contained by: file stellar_populations.spectra.postprocess.recent.F90
Modules used: input_parameters

function: recentgenericconstructor
Description: Generic constructor for the recent spectrum postprocessor class.
Code lines: 8
Contained by: file stellar_populations.spectra.postprocess.recent.F90
19. Source Code Documentation

interface: spectrapostprocessorrecent
Description: Constructors for the recent spectrum postprocessor class.
Code lines: 4
Contained by: file stellar_populations.spectra.postprocess.recent.F90

file: structure_formation.cosmological_mass_variance.filtered_power_spectrum.F90
Description: Contains a module which implements calculation of \( \sigma(M) \) via filtering of the power spectrum.
Code lines: 210

module: cosmological_mass_variance_filtered_power_spectrum
Description: Implements calculation of \( \sigma(M) \) via filtering of the power spectrum.
Code lines: 190
Contained by: file structure_formation.cosmological_mass_variance.filtered_power_spectrum.F90
Used by: subroutine galacticus_state_retrieve subroutine galacticus_state_store subroutine initialize_cosmological_mass_variance

subroutine: cosmological_mass_variance_filtered_power_spectrum_initialize
Description: Initializes the \( \sigma(M) \) calculation for the “filtered power spectrum” method.
Code lines: 34
Contained by: module cosmological_mass_variance_filtered_power_spectrum Modules used: input_parameters iso_varying_string

subroutine: cosmological_mass_variance_filtered_power_spectrum_tabulate
Description: Tabulate the virial density contrast for the Kitayama and Suto [1996] fitting function module.
Code lines: 37
Contained by: module cosmological_mass_variance_filtered_power_spectrum Modules used: tables

subroutine: cosmological_mass_variance_fps_state_retrieve
Description: Retrieve the tabulation state from the file.
Code lines: 10
Contained by: module cosmological_mass_variance_filtered_power_spectrum Modules used: fgsl

subroutine: cosmological_mass_variance_fps_state_store
Description: Write the tabulation state to file.
Code lines: 9
Contained by: module cosmological_mass_variance_filtered_power_spectrum Modules used: fgsl

function: variance_integral
Description: Compute the root-variance of mass in spheres enclosing the given mass from the power spectrum.
Code lines: 30
Contained by: module cosmological_mass_variance_filtered_power_spectrum Modules used: cosmology_parameters iso_c_binding numerical_constants_math numerical_integration
19.1. Program units

```plaintext
power_spectrum_window_functions

function: variance_integrand
Description: Integrand function used in compute the variance in (real space) top-hat spheres from the power spectrum.
Code lines: 14
Contained by: module cosmological_mass_variance_filtered_power_spectrum
Modules used: iso_c_binding power_spectrum_window_functions

function: variance_integrand_tophat
Description: Integrand function used in compute the variance in (real space) top-hat spheres from the power spectrum.
Code lines: 14
Contained by: module cosmological_mass_variance_filtered_power_spectrum
Modules used: iso_c_binding

file: structure_formation.critical_overdensity.F90
Description: Contains a module which implements the critical linear theory overdensity for halo collapse.
Code lines: 393

module: critical_overdensity
Description: Implements the critical linear theory overdensity for halo collapse.
Code lines: 373
Contained by: file structure_formation.critical_overdensity.F90
Modules used: iso_varying_string tables
Used by: function expansion_factor_at_formation function dark_matter_halo_mass_accretion_time_zhao2009
function growthrateodes function diemerkravtsov2014concentration
function diemerkravtsov2014concentration function nfw1996concentration
function nfw1996concentration function dark_matter_profile_shape_gao2008
subroutine galacticus_state_retrieve subroutine halo_mass_function_compute
calculate common_factors subroutine merger_tree_build_dcole2000
subroutine merger_tree_regrid_time function virial_orbital_parameters_wetzel2010
function excursion_sets_barrier_critical overdensity function excursion_sets_barrier_gradient_critical_overdensity
function dark_matter_halo_bias_press_schechter function dark_matter_halo_bias_smt
tinker2010 function halo_mass_function_sheth_tormen_differential
```
function: collapsing_mass_root
Description: Function used in finding the mass of halo just collapsing at a given cosmic epoch.
Code lines: 7
Contained by: module critical_overdensity
Modules used: power_spectra

function: critical_overdensity_collapsing_mass
Description: Return the mass scale just collapsing at the given cosmic time.
Code lines: 28
Contained by: module critical_overdensity
Modules used: cosmology_functions root_finder

function: critical_overdensity_forCollapse
Description: Return the linear theory critical overdensity for collapse at the given cosmic time.
Code lines: 47
Contained by: module critical_overdensity
Modules used: cosmology_functions galacticus_error

function: critical_overdensity_forCollapse_timeGradient
Description: Return the derivative with respect to time of the linear theory critical overdensity for collapse at the given cosmic time.
Code lines: 47
Contained by: module critical_overdensity
Modules used: cosmology_functions galacticus_error

subroutine: critical_overdensity_initialize
Description: Initializes the critical overdensity module.
Code lines: 64
Contained by: module critical_overdensity
Modules used: array_utilities critical_overdensities_kitayama_-suto1996 galacticus_error input_parameters sphericalCollapse_matter_dark_energy sphericalCollapse_matter_lambda

function: critical_overdensity_massScaling
Description: Return a multiplicative, mass-dependent factor by which the critical overdensity should be scaled.
Code lines: 11
Contained by: module critical_overdensity

function: critical_overdensity_massScaling_gradient
Description: Return the gradient with mass of a multiplicative, mass-dependent factor by which the critical overdensity should be scaled.
19.1. Program units

**Code lines:** 11  
**Contained by:** module `critical_overdensity`

**subroutine:** `critical_overdensity_mass_scaling_initialize`  
**Description:** Initializes the critical overdensity mass scaling method.  
**Code lines:** 54  
**Contained by:** module `critical_overdensity`  
**Modules used:**  
- `critical_overdensity_mass_scalings_-null`
- `critical_overdensity_mass_scalings_-wdm`
- `galacticus_error`
- `input_parameters`

**subroutine:** `critical_overdensity_state_retrieve`  
**Description:** Reset the tabulation if state is to be retrieved. This will force tables to be rebuilt.  
**Code lines:** 9  
**Contained by:** module `critical_overdensity`  
**Modules used:** `fgsl`

**function:** `time_ofCollapse`  
**Description:** Returns the time of collapse for a perturbation of linear theory overdensity `criticalOverdensity`.  
**Code lines:** 31  
**Contained by:** module `critical_overdensity`  
**Modules used:** `cosmology_functions`

**file:** `structure_formation.critical_overdensity.Kitayama_Suto1996.F90`  
**Description:** Contains a module which implements calculations of critical overdensity using the fitting function of Kitayama and Suto [1996].  
**Code lines:** 95

**module:** `critical_overdensities_kitayama_suto1996`  
**Code lines:** 74  
**Contained by:** file `structure_formation.critical_overdensity.Kitayama_Suto1996.F90`  
**Used by:** subroutine `critical_overdensity_-initialize`

**subroutine:** `critical_overdensity_kitayama_suto1996`  
**Description:** Tabulate the virial density contrast for the Kitayama and Suto [1996] fitting function module.  
**Code lines:** 37  
**Contained by:** module `critical_overdensities_kitayama_suto1996`  
**Modules used:**  
- `cosmology_functions`
- `linear_growth`
- `numerical_constants_math`
- `tables`

**subroutine:** `critical_overdensity_kitayama_suto1996_initialize`  
**Description:** Initializes the $\delta_c$ calculation for the Kitayama and Suto [1996] fitting function module.  
**Code lines:** 19  
**Contained by:** module `critical_overdensities_kitayama_suto1996`  
**Modules used:**  
- `cosmology_parameters`
- `galacticus_error`
- `iso_varying_string`
- `numerical_comparison`
19. Source Code Documentation

file: structure_formation.critical_overdensity.mass_scaling.null.F90
Description: Contains a module which implements a null scaling of critical overdensities for collapse.
Code lines: 64

module: critical_overdensity_mass_scalings_null
Description: Implements a null scaling of critical overdensities for collapse.
Code lines: 44
Contained by: file structure_formation.critical_overdensity.mass_scaling.null.F90
Used by: subroutine critical_overdensity_mass_scaling_initialize

function: critical_overdensity_mass_scaling_gradient_null
Description: Returns the gradient of a mass scaling for critical overdensities that is always unity.
Code lines: 7
Contained by: module critical_overdensity_mass_scalings_null

function: critical_overdensity_mass_scaling_null
Description: Returns a mass scaling for critical overdensities that is always unity.
Code lines: 7
Contained by: module critical_overdensity_mass_scalings_null

subroutine: critical_overdensity_mass_scaling_null_initialize
Description: Initializes the “null” critical overdensity mass scaling method.
Code lines: 13
Contained by: module critical_overdensity_mass_scalings_null
Modules used: iso_varying_string

file: structure_formation.critical_overdensity.mass_scaling.warm_dark_matter.F90
Description: Contains a module which implements a warm dark matter scaling of critical overdensities for collapse based on the work of Barkana et al. [2001].
Code lines: 207

module: critical_overdensity_mass_scalings_wdm
Description: Implements a warm dark matter scaling of critical overdensities for collapse based on the work of Barkana et al. [2001].
Code lines: 187
Contained by: file structure_formation.critical_overdensity.mass_scaling.warm_dark_matter.F90
Modules used: fgsl
Used by: subroutine critical_overdensity_mass_scaling_initialize

function: critical_overdensity_mass_scaling_gradient_wdm
19.1. Program units

**Description:** Returns a mass scaling for critical overdensities based on the results of Barkana et al. [2001]. This method assumes that their results for the original collapse barrier (i.e. the critical overdensity, and which they call $B_0$) scale with the effective Jeans mass of the warm dark matter particle as computed using their eqn. (10).

**Code lines:** 31

**Contained by:** module `critical_overdensity_mass_scalings_wdm`

**Modules used:** `numerical_interpolation`
function: critical_overdensity_mass_scaling_wdm
  Description: Returns a mass scaling for critical overdensities based on the results of Barkana et al. [2001]. This method assumes that their results for the original collapse barrier (i.e. the critical overdensity, and which they call $B_0$) scale with the effective Jeans mass of the warm dark matter particle as computed using their eqn. (10).
  Code lines: 34
  Contained by: module critical_overdensity_mass_scalings_wdm
  Modules used: numerical_interpolation

subroutine: critical_overdensity_mass_scaling_wdm_initialize
  Description: Initializes the “warmDarkMatter” critical overdensity mass scaling method.
  Code lines: 80
  Contained by: module critical_overdensity_mass_scalings_wdm
  Modules used: cosmology_parameters fox_dom galacticus_error galacticus_input_paths input_parameters io_xml iso_varying_string

file: structure_formation.excursion_sets.barrier.F90
  Description: Contains a module which implements calculations of barriers for excursion set calculations.
  Code lines: 335

module: excursion_sets_barriers
  Description: Implements calculations of barriers for excursion set calculations.
  Code lines: 314
  Contained by: file structure_formation.excursion_sets.barrier.F90
  Modules used: iso_varying_string
  Used by: program tests_excursion_sets function excursion_sets_barrier_effective function excursion_sets_first_crossing_probability_farahi function g_1 function g_2 function g_2_integrated function g_2_integrated function excursion_sets_barrier_effective function excursion_sets_first_crossing_probability_linear

function: excursion_sets_barrier
  Description: Return the barrier for excursion sets at the given variance and time.
  Code lines: 50
  Contained by: module excursion_sets_barriers
  Modules used: excursion_sets_barriers_remap_null excursion_sets_barriers_remap_scale excursion_sets_barriers_remap_smt

function: excursion_sets_barrier_gradient
  Description: Return the gradient (with respect to mass) of the barrier for excursion sets at the given variance and time.
  Code lines: 51
  Contained by: module excursion_sets_barriers
Modules used: excursion_sets_barriers_remap_null excursion_sets_barriers_remap_scale excursion_sets_barriers_remap_smt

subroutine: excursion_sets_barrier_initialize
Description: Initialize the excursion sets barrier module.
Code lines: 171
Contained by: module excursion_sets_barriers
Modules used: excursion_sets_barriers_critical_overdensity excursion_sets_barriers_linear
excursion_sets_barriers_quadratic excursion_sets_barriers_remap_null excursion_sets_barriers_remap_scale excursion_sets_barriers_remap_smt
galacticus_error input_parameters memory_management string_handling

function: excursion_sets_barrier_name
Description: Return the fully-qualified name of the selected excursion set barrier.
Code lines: 6
Contained by: module excursion_sets_barriers

file: structure_formation.excursion_sets.barrier.critical_overdensity.F90
Description: Contains a module which implements a barrier for excursion set calculations of dark matter halo formation which equals the critical overdensity for collapse.
Code lines: 92

module: excursion_sets_barriers_critical_overdensity
Description: Implements a barrier for excursion set calculations of dark matter halo formation which equals the critical overdensity for collapse.
Code lines: 71
Contained by: file structure_formation.excursion_sets.barrier.critical_overdensity.F90
Used by: subroutine excursion_sets_barrier_initialize

function: excursion_sets_barrier_critical_overdensity
Description: Return a critical overdensity barrier for excursion set calculations at the given variance.
Code lines: 18
Contained by: module excursion_sets_barriers_critical_overdensity
Modules used: critical_overdensity power_spectra

function: excursion_sets_barrier_gradient_critical_overdensity
Description: Return the gradient of a critical overdensity barrier for excursion set calculations at the given variance.
Code lines: 20
Contained by: module excursion_sets_barriers_critical_overdensity
Modules used: critical_overdensity power_spectra

subroutine: excursion_sets_barriers_critical_overdensity.Initialize
Description: Initialize the critical overdensity excursion set barrier module.
Code lines: 16
Contained by: module excursion_sets_barriers_critical_overdensity
19. Source Code Documentation

Modules used: iso_varying_string

file: structure_formation.excursion_sets.barrier.linear.F90
Description: Contains a module which implements a linear barrier for excursion set calculations of dark matter halo formation.
Code lines: 96

module: excursion_sets_barriers_linear
Description: Implements a linear barrier for excursion set calculations of dark matter halo formation.
Code lines: 76
Contained by: file structure_formation.excursion_sets.barrier.linear.F90
Used by: subroutine excursion_sets_barrier_initialize

function: excursion_sets_barrier_gradient_linear
Description: Return the gradient of a linear barrier for excursion set calculations at the given variance.
Code lines: 7
Contained by: module excursion_sets_barriers_linear

function: excursion_sets_barrier_linear
Description: Return a linear barrier for excursion set calculations at the given variance.
Code lines: 7
Contained by: module excursion_sets_barriers_linear

subroutine: excursion_sets_barriers_linear_initialize
Description: Initialize the linear excursion set barrier module.
Code lines: 43
Contained by: module excursion_sets_barriers_linear
Modules used: input_parameters iso_varying_string

file: structure_formation.excursion_sets.barrier.quadratic.F90
Description: Contains a module which implements a quadratic barrier for excursion set calculations of dark matter halo formation.
Code lines: 109

module: excursion_sets_barriers_quadratic
Description: Implements a quadratic barrier for excursion set calculations of dark matter halo formation.
Code lines: 89
Contained by: file structure_formation.excursion_sets.barrier.quadratic.F90
Used by: subroutine excursion_sets_barrier_initialize

function: excursion_sets_barrier_gradient_quadratic
Description: Return the gradient of a quadratic barrier for excursion set calculations at the given variance.
Code lines: 7
Contained by: module excursion_sets_barriers_quadratic

function: excursion_sets_barrier_quadratic
Description: Return a quadratic barrier for excursion set calculations at the given variance.
19.1. Program units

Code lines: 7
Contained by: module excursion_sets_barriers_quadratic

subroutine: excursion_sets_barriers_quadratic_initialize
Description: Initialize the quadratic excursion set barrier module.
Code lines: 56
Contained by: module excursion_sets_barriers_quadratic
Modules used: input_parameters iso_varying_string

file: structure_formation.excursion_sets.barrier.remap.Sheth-Mo-Tormen.F90
Description: Contains a module which implements a Sheth et al. [2001] remapping of excursion set barriers.
Code lines: 107

module: excursion_sets_barriers_remap_smt
Description: Implements a Sheth et al. [2001] remapping of excursion set barriers.
Code lines: 87
Contained by: file structure_formation.excursion_sets.barrier.remap.Sheth-Mo-Tormen.F90
Used by: subroutine excursion_sets_barrier_gradient_remap_smt

subroutine: excursion_sets_barrier_gradient_remap_smt
Description: Return the gradient of the barrier for excursion set calculations remapped according to Sheth et al. [2001].
Code lines: 16
Contained by: module excursion_sets_barriers_remap_smt

subroutine: excursion_sets_barrier_remap_smt
Description: Return the barrier for excursion set calculations remapped according to Sheth et al. [2001].
Code lines: 10
Contained by: module excursion_sets_barriers_remap_smt

subroutine: excursion_sets_barriers_remap_smt_initialize
Description: Initialize the Sheth et al. [2001] excursion set barrier remapping module.
Code lines: 31
Contained by: module excursion_sets_barriers_remap_smt
Modules used: iso_varying_string

file: structure_formation.excursion_sets.barrier.remap.null.F90
Description: Contains a module which implements a null remapping of excursion set barriers.
Code lines: 85

module: excursion_sets_barriers_remap_null
Description: Implements a null remapping of excursion set barriers.
Code lines: 65
Contained by: file structure_formation.excursion_sets.barrier.remap.null.F90
Used by: function excursion_sets_barrier

subroutine excursion_sets_barrier_gradient_remap_null

Description: Return the gradient of the barrier for excursion set calculations unmodified.
Code lines: 9
Contained by: module excursion_sets_barriers_remap_null

subroutine: excursion_sets_barrier_remap_null

Description: Return the barrier for excursion set calculations unmodified.
Code lines: 9
Contained by: module excursion_sets_barriers_remap_null

subroutine: excursion_sets_barriers_remap_null_initialize

Description: Initialize the null excursion set barrier remapping module.
Code lines: 22
Contained by: module excursion_sets_barriers_remap_null
Modules used: iso_varying_string

file: structure_formation.excursion_sets.barrier.remap.scale.F90

Description: Contains a module which implements a scale remapping of excursion set barriers.
Code lines: 116

module: excursion_sets_barriers_remap_scale

Description: Implements a scaling remapping of excursion set barriers.
Code lines: 96
Contained by: file structure_formation.excursion_sets.barrier.remap.scale.F90
Used by: function excursion_sets_barrier    function excursion_sets_barrier_gradient
          subroutine excursion_sets_barrier_gradient_remap_null

subroutine: excursion_sets_barrier_gradient_remap_scale

Description: Return the gradient of the barrier for excursion set calculations unmodified.
Code lines: 10
Contained by: module excursion_sets_barriers_remap_scale

subroutine: excursion_sets_barrier_remap_scale

Description: Return the barrier for excursion set calculations unmodified.
Code lines: 10
Contained by: module excursion_sets_barriers_remap_scale

subroutine: excursion_sets_barriers_remap_scale_initialize

Description: Initialize the scale excursion set barrier remapping module.
Code lines: 47
Contained by: module excursion_sets_barriers_remap_scale
Modules used: input_parameters    iso_varying_string
19.1. Program units

file: structure_formation.excursion_sets.first_crossing_distribution.F90
Description: Contains a module which implements calculations of first crossing distributions for excursion set calculations.
Code lines: 134

module: excursion_sets_first_crossings
Description: Implements calculations of first crossing distributions for excursion set calculations.
Code lines: 113
Contained by: file structure_formation.excursion_sets.first_crossing_distribution.F90
Modules used: iso_varying_string
Used by: program tests_excursion_sets subroutine excursion_sets_maximum_-sigma_test
function generalized_press_schechter_- subresolution_fraction
function halo_mass_function_- differential_press_schechter

function: excursion_sets_first_crossing_probability
Description: Return the probability of first crossing for excursion sets at the given variance and time.
Code lines: 10
Contained by: module excursion_sets_first_crossings

function: excursion_sets_first_crossing_rate
Description: Return the rate of first crossing for excursion sets beginning at the given variance and time to transition to a first crossing at the given varianceProgenitor.
Code lines: 10
Contained by: module excursion_sets_first_crossings

subroutine: excursion_sets_first_crossings_initialize
Description: Initialize the excursion sets first crossing distribution module.
Code lines: 56
Contained by: module excursion_sets_first_crossings
Modules used: excursion_sets_first_crossing_farahi excursion_sets_first_crossing_-linear_barrier
excursion_sets_first_crossing_zhang_-hui excursion_sets_first_crossing_zhang_-hui_high
galacticus_error input_parameters

function: excursion_sets_non_crossing_rate
Description: Return the rate of non-crossing for excursion sets beginning at the given variance and time.
Code lines: 10
Contained by: module excursion_sets_first_crossings

file: structure_formation.excursion_sets.first_crossing_distribution.Farahi.F90
Description: Contains a module which implements a fast and accurate method to solve the excursion set barrier crossing problem for generic barriers.
Code lines: 667

module: excursion_sets_first_crossing_farahi
Description: Implements a fast and accurate method to solve the excursion set barrier crossing problem for generic barriers.

Code lines: 643
Contained by: file `structure_formation.excursion_sets.first_crossing_distribution.Farahi.F90`
Modules used: `fgsl`, `iso_c_binding`, `iso_varying_string`
Used by: subroutine `excursion_sets_first_crossings_initialize`;

function: `erfapproximation`
Description: An approximation to the error function that is designed to be very accurate in the vicinity of zero and infinity.

Code lines: 17
Contained by: module `excursion_sets_first_crossing_farahi`
Modules used: `numerical_constants_math`

function: `excursion_sets_barrier_effective`
Description: The effective barrier for conditional excursion sets.

Code lines: 18
Contained by: module `excursion_sets_first_crossing_farahi`
Modules used: `excursion_sets_barriers`, `kind_numbers`

subroutine: `excursion_sets_first_crossing_farahi_initialize`
Description: Initialize the “Farahi” first crossing distribution method for excursion sets module.

Code lines: 40
Contained by: module `excursion_sets_first_crossing_farahi`
Modules used: `input_parameters`

subroutine: `excursion_sets_first_crossing_farahi_read_file`
Description: Read tabulated data on excursion set first crossing probabilities from file.

Code lines: 88
Contained by: module `excursion_sets_first_crossing_farahi`
Modules used: `file_utilities`, `io_hdf5`, `memory_management`, `numerical_interpolation`

subroutine: `excursion_sets_first_crossing_farahi_write_file`
Description: Write tabulated data on excursion set first crossing probabilities to file.

Code lines: 32
Contained by: module `excursion_sets_first_crossing_farahi`
Modules used: `hdf5`, `io_hdf5`

function: `excursion_sets_first_crossing_probability_farahi`
Description: Return the probability for excursion set first crossing using the methodology of Farahi.

Code lines: 110
Contained by: module `excursion_sets_first_crossing_farahi`
Modules used: `excursion_sets_barriers`, `galacticus_display`, `kind_numbers`, `memory_management`, `numerical_interpolation`, `numerical_ranges`
function: excursion_sets_first_crossing_rate_farahi

Description: Return the rate for excursion set first crossing.

Code lines: 62

Contained by: module excursion_sets_first_crossing_farahi

Modules used: numerical_interpolation

subroutine: excursion_sets_first_crossing_rate_tabulate_farahi

Description: Tabulate the excursion set crossing rate.

Code lines: 158

Contained by: module excursion_sets_first_crossing_farahi

Modules used: cosmology_functions galacticus_display kind_numbers memory_management numerical_interpolation numerical_ranges

function: excursion_sets_non_crossing_rate_farahi

Description: Return the rate for excursion set non-crossing.

Code lines: 34

Contained by: module excursion_sets_first_crossing_farahi

Modules used: numerical_interpolation

function: make_variance_range

Description: Builds a numerical range between rangeMinimum and rangeMaximum using rangeNumber points with spacing that varies from logarithmic to linear spacing with the transition point controlled by ratioAtMaximum.

Code lines: 16

Contained by: module excursion_sets_first_crossing_farahi

file: structure_formation.excursion_sets.first_crossing_distribution.Zhang_Hui.F90

Description: Contains a module which implements the first crossing distribution for excursion set calculations of dark matter halo formation using the methodology of Zhang and Hui [2006].

Code lines: 171

module: excursion_sets_first_crossing_zhang_hui

Description: Implements the first crossing distribution for excursion set calculations of dark matter halo formation using the methodology of Zhang and Hui [2006].

Code lines: 150

Contained by: file structure_formation.excursion_sets.first_crossing_distribution.Zhang_Hui.F90

Modules used: fgsl

Used by: subroutine excursion_sets_first_crossings_initialize

function: excursion_sets_first_crossing_probability_zhang_hui

Description: Return the probability for excursion set first crossing using the methodology of Zhang and Hui [2006].

Code lines: 89

Contained by: module excursion_sets_first_crossing_zhang_hui

Modules used: excursion_sets_first_crossing_zhang_hui galacticus_display hui_utilities
19. Source Code Documentation

```
isoc_binding                      memory_management
numerical_interpolation          numerical_ranges

function: excursion_sets_first_crossing_rate_zhang_hui
Description: Return the rate for excursion set first crossing.
Code lines: 8
Contained by: module excursion_sets_first_crossing_zhang_hui
Modules used: galacticus_error

subroutine: excursion_sets_first_crossing_zhang_hui_initialize
Description: Initialize the “ZhangHui2006” first crossing distribution for excursion sets module.
Code lines: 15
Contained by: module excursion_sets_first_crossing_zhang_hui
Modules used: iso_varying_string

function: excursion_sets_non_crossing_rate_zhang_hui
Description: Return the rate for excursion set non-crossing.
Code lines: 8
Contained by: module excursion_sets_first_crossing_zhang_hui
Modules used: galacticus_error

Description: Contains a module which implements various utility functions for excursion set first crossing
distribution set calculations that use variants of the methodology of Zhang and Hui [2006].
Code lines: 147

module: excursion_sets_first_crossing_zhang_hui_utilities
Description: Implements various utility functions for excursion set first crossing distribution set calculations that use variants of the methodology of Zhang and Hui [2006].
Code lines: 126
Contained by: file structure_formation.excursion_sets.first_crossing_distribution.Zhang_Hui.utilities.F90
Used by: function excursion_sets_first_crossing_probability_zhang_hui
         function excursion_sets_first_crossing_probability_zhang_hui_high

function: delta
Description: Returns the factor $\Delta_{i,j}$ in the Zhang and Hui [2006] algorithm for excursion set barrier
crossing probabilities.
Code lines: 22
Contained by: module excursion_sets_first_crossing_zhang_hui_utilities

function: g_1
Description: Returns the function $g_1(S)$ in the Zhang and Hui [2006] algorithm for excursion set barrier
crossing probabilities.
Code lines: 11
Contained by: module excursion_sets_first_crossing_zhang_hui_utilities
Modules used: excursion_sets_barriers

function: g_2
Description: Returns the function $g_2(S, S')$ in the Zhang and Hui [2006] algorithm for excursion set barrier
crossing probabilities.
```
19.1. Program units

Code lines: 20
Contained by: module excursion_sets_first_crossing_zhang_hui_utilities
Modules used: excursion_sets_barriers math_distributions_gaussian

function: g_2_integrand_zhang_hui
Description: Integrand function used in computing $\Delta_{i,i}$ in the Zhang and Hui [2006] algorithm for excursion set barrier crossing probabilities.
Code lines: 19
Contained by: module excursion_sets_first_crossing_zhang_hui_utilities
Modules used: excursion_sets_barriers iso_c_binding
math_distributions_gaussian

function: g_2_integrated
Code lines: 32
Contained by: module excursion_sets_first_crossing_zhang_hui_utilities
Modules used: excursion_sets_barriers f gsl
iso_c_binding numerical_comparison
numerical_integration

file: structure_formation.excursion_sets.first_crossing_distribution.Zhang_Hui_high_order.F90
Description: Contains a module which implements the first crossing distribution for excursion set calculations of dark matter halo formation using a high order modification of the methodology of Zhang and Hui [2006].
Code lines: 218

module: excursion_sets_first_crossing_zhang_hui_high
Description: Implements the first crossing distribution for excursion set calculations of dark matter halo formation using a high order modification of the methodology of Zhang and Hui [2006].
Code lines: 195
Contained by: file structure_formation.excursion_sets.first_crossing_distribution.Zhang_Hui_high_order.F90
Modules used: f gsl
Used by: subroutine excursion_sets_first_crossings_initialize

function: excursion_sets_first_crossing_probability_zhang_hui_high
Description: Return the probability for excursion set first crossing using a high order modification of the methodology of Zhang and Hui [2006].
Code lines: 134
Contained by: module excursion_sets_first_crossing_zhang_hui_high
Modules used: excursion_sets_first_crossing_zhang_hui_utilities galacticus_display
hui_utilities iso_c Binding memory_management
numerical_interpolation numerical_ranges

function: excursion_sets_first_crossing_rate_zhang_hui_high
Description: Return the rate for excursion set first crossing assuming a linear barrier.
Code lines: 8
Contained by: module excursion_sets_first_crossing_zhang_hui_high
Modules used: galacticus_error
**subroutine: excursion_sets_first_crossing_zhang_hui_high_initialize**
*Description:* Initialize the “ZhangHui2006 high order” first crossing distribution for excursion sets module.
*Code lines:* 15
*Contained by:* module excursion_sets_first_crossing_zhang_hui_high
*Modules used:* iso_varying_string

**function: excursion_sets_non_crossing_rate_zhang_hui_high**
*Description:* Return the rate for excursion set non-crossing.
*Code lines:* 8
*Contained by:* module excursion_sets_first_crossing_zhang_hui_high
*Modules used:* galacticus_error

**file: structure_formation.excursion_sets.first_crossing_distribution.linear_barrier.F90**
*Description:* Contains a module which implements the first crossing distribution (assuming a linear barrier) for excursion set calculations of dark matter halo formation.
*Code lines:* 103

**module: excursion_sets_first_crossing_linear_barrier**
*Description:* Implements the first crossing distribution (assuming a linear barrier) for excursion set calculations of dark matter halo formation.
*Code lines:* 82
*Contained by:* file structure_formation.excursion_sets.first_crossing_distribution.linear_barrier.F90
*Used by:* subroutine excursion_sets_first_crossings_initialize

**function: excursion_sets_barrier_effective**
*Description:* The effective barrier for conditional excursion sets.
*Code lines:* 8
*Contained by:* module excursion_sets_first_crossing_linear_barrier
*Modules used:* excursion_sets_barriers

**subroutine: excursion_sets_first_crossing_linear_barrier_initialize**
*Description:* Initialize the linear barrier first crossing distribution for excursion sets module.
*Code lines:* 15
*Contained by:* module excursion_sets_first_crossing_linear_barrier
*Modules used:* iso_varying_string

**function: excursion_sets_first_crossing_probability_linear**
*Description:* Return the probability for excursion set first crossing assuming a linear barrier. Uses the analytic solution for this case Sheth [1998], Sheth and Tormen [2002].
*Code lines:* 10
*Contained by:* module excursion_sets_first_crossing_linear_barrier
*Modules used:* excursion_sets_barriers numerical_constants_math

**function: excursion_sets_first_crossing_rate_linear**
19.1. Program units

**Description:** Return the rate for excursion set first crossing assuming a linear barrier. Uses the analytic solution for this case Sheth [1998], Sheth and Tormen [2002] with a simple offset in the starting coordinates. The rate of barrier crossing is computed by solving for the first crossing distribution at a slightly earlier time and then dividing through by that time interval.

**Code lines:** 19

**Contained by:** module excursion_sets_first_crossing_linear_barrier
19. Source Code Documentation

Modules used: numerical_constants_math

function: excursion_sets_non_crossing_rate_linear
Description: Return the rate for excursion set non-crossing assuming a linear barrier. For a linear barrier the integral over the crossing probability (from zero to infinite variance) equals unity, so all trajectories cross. The non-crossing rate is therefore zero.

Code lines: 9
Contained by: module excursion_sets_first_crossing_linear_barrier

file: structure_formation.halo_bias.F90
Description: Contains a module which implements calculations of dark matter halo bias.
Code lines: 142

module: dark_matter_halo_biases
Description: Implements calculations of dark matter halo bias.
Code lines: 122
Contained by: file structure_formation.halo_bias.F90
Modules used: galacticus_nodes iso_varying_string
Used by: subroutine galacticus_output_halo_model subroutine halo_mass_function_compute function bias_integrand_i

interface: dark_matter_halo_bias
Code lines: 3
Contained by: module dark_matter_halo_biases

function: dark_matter_halo_bias_by_mass
Description: Computes the bias for a dark matter halo.
Code lines: 12
Contained by: module dark_matter_halo_biases

function: dark_matter_halo_bias_by_node
Description: Computes the bias for a dark matter halo.
Code lines: 12
Contained by: module dark_matter_halo_biases

subroutine: dark_matter_halo_bias_initialize
Description: Initialize the dark matter halo bias module.
Code lines: 56
Contained by: module dark_matter_halo_biases
Modules used: dark_matter_halo_biases_press_schechter dark_matter_halo_biases_smt
schechter
dark_matter_halo_biases_tinker2010 galacticus_error
input_parameters

file: structure_formation.halo_bias.Press-Schechter.F90
Description: Contains a module which implements calculations of halo bias using the Press-Schechter mass function [Mo and White, 1996].
Code lines: 77

module: dark_matter_halo_biases_press_schechter
19.1. Program units

Description: Implements calculations of halo bias using the Press-Schechter mass function [Mo and White, 1996].

Code lines: 57

Contained by: file structure_formation.halo_bias.Press-Schechter.F90

Used by: subroutine dark_matter_halo_bias_-initialize

function: dark_matter_halo_bias_node_press_schechter
Description: Computes the bias for a dark matter halo using the method of Mo and White [1996].

Code lines: 11

Contained by: module dark_matter_halo_biases_press_schechter

Modules used: galacticus_nodes

function: dark_matter_halo_bias_press_schechter
Description: Computes the bias for a dark matter halo using the method of Mo and White [1996].

Code lines: 16

Contained by: module dark_matter_halo_biases_press_schechter

Modules used: critical_overdensity power_spectra

subroutine: dark_matter_halo_bias_press_schechter_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.

Code lines: 13

Contained by: module dark_matter_halo_biases_press_schechter

Modules used: iso_varying_string

file: structure_formation.halo_bias.SMT.F90
Description: Contains a module which implements calculations of halo bias using the fitting function of Sheth et al. [2001].

Code lines: 78

module: dark_matter_halo_biases_smt
Description: Implements calculations of halo bias using the fitting function of Sheth et al. [2001].

Code lines: 58

Contained by: file structure_formation.halo_bias.SMT.F90

Used by: subroutine dark_matter_halo_bias_-initialize

function: dark_matter_halo_bias_node_smt
Description: Computes the bias for a dark matter halo using the method of Mo and White [1996].

Code lines: 11

Contained by: module dark_matter_halo_biases_smt

Modules used: galacticus_nodes

function: dark_matter_halo_bias_smt
Description: Computes the bias for a dark matter halo using the method of Sheth et al. [2001].

Code lines: 17

Contained by: module dark_matter_halo_biases_smt

Modules used: critical_overdensity power_spectra
subroutine: dark_matter_halo_bias_smt_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 13
Contained by: module dark_matter_halo_biases_smt
Modules used: iso_varying_string

file: structure_formation.halo_bias.Tinker2010.F90
Description: Contains a module which implements calculations of halo bias using the fitting function of Tinker et al. [2010].
Code lines: 98

module: dark_matter_halo_biases_tinker2010
Description: Implements calculations of halo bias using the fitting function of Tinker et al. [2010].
Code lines: 78
Contained by: file structure_formation.halo_bias.Tinker2010.F90
Used by: subroutine dark_matter_halo_bias_smt_initialize

function: dark_matter_halo_bias_node_tinker2010
Description: Computes the bias for a dark matter halo using the method of Mo and White [1996].
Code lines: 11
Contained by: module dark_matter_halo_biases_tinker2010
Modules used: galacticus_nodes

function: dark_matter_halo_bias_tinker2010
Description: Computes the bias for a dark matter halo using the method of Tinker et al. [2010].
Code lines: 37
Contained by: module dark_matter_halo_biases_tinker2010
Modules used: critical_overdensity power_spectra virial_density_contrast

subroutine: dark_matter_halo_bias_tinker2010_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 13
Contained by: module dark_matter_halo_biases_tinker2010
Modules used: iso_varying_string

file: structure_formation.halo_mass_function.F90
Code lines: 215

module: halo_mass_function
Code lines: 197
Contained by: file structure_formation.halo_mass_function.F90
Modules used: fgs1 iso_c_binding iso_varying_string
Used by: function mass_function_halo_mass_integrand program tests_excursion_sets
subroutine halo_mass_function_compute
function merger_tree_construct_mass_-function_sampling_halo_mf
subroutine galacticustreeindicesread
program optimal_sampling_smf
function mass_function_integrand_i

function: halo_mass_fraction_integrand
Description: Integrand function used in computing the halo mass fraction.
Code lines: 17
Contained by: module halo_mass_function

function: halo_mass_fraction_integrated
Description: Return the halo mass fraction integrated between massLow and massHigh.
Code lines: 24
Contained by: module halo_mass_function
Modules used: cosmology_parameters numerical_integration

function: halo_mass_function_differential
Description: Return the differential halo mass function for mass \([M_{\odot}]\) at time.
Code lines: 36
Contained by: module halo_mass_function
Modules used: halo_mass_function_modifiers_simple_-systematic

subroutine: halo_mass_function_initialize
Description: Initializes the halo mass function module.
Code lines: 56
Contained by: module halo_mass_function
Modules used: galacticus_error halo_mass_function_press_schechter
halo_mass_function_sheth_tormen halo_mass_function_tinker2008
input_parameters

function: halo_mass_function_integrand
Code lines: 15
Contained by: module halo_mass_function

function: halo_mass_function_integrated
Description: Return the halo mass function integrated between massLow and massHigh.
Code lines: 18
Contained by: module halo_mass_function
Modules used: numerical_integration

file: structure_formation.halo_mass_function.Press-Schechter.F90
Description: Contains a module which generates a tabulated Press-Schechter halo mass function.
Code lines: 63

module: halo_mass_function_press_schechter
19. Source Code Documentation

**Description:** Implements generation of a tabulated power-law primordial power spectrum.

**Code lines:** 43

**Contained by:** file `structure_formation.halo_mass_function.Press-Schechter.F90`

**Used by:** subroutine `halo_mass_function_-
  initialize`

**function:** `halo_mass_function_differential_press_schechter`

**Description:** Compute the Press-Schechter halo mass function.

**Code lines:** 16

**Contained by:** module `halo_mass_function_press_schechter`

**Modules used:** `cosmology_parameters excursion_sets_first_crossings power_spectra`

**subroutine:** `halo_mass_function_press_schechter_initialize`

**Description:** Initializes the “Press-Schechter mass function” module.

**Code lines:** 9

**Contained by:** module `halo_mass_function_press_schechter`

**Modules used:** `iso_varying_string`

**file:** `structure_formation.halo_mass_function.Sheth-Tormen.F90`

**Description:** Contains a module which generates a tabulated Sheth-Tormen halo mass function.

**Code lines:** 67

**module:** `halo_mass_function_sheth_tormen`

**Description:** Implements generation of a tabulated power-law primordial power spectrum.

**Code lines:** 47

**Contained by:** file `structure_formation.halo_mass_function.Sheth-Tormen.F90`

**Used by:** subroutine `halo_mass_function_-
  initialize`

**function:** `halo_mass_function_sheth_tormen_differential`

**Description:** Compute the Sheth-Tormen halo mass function.

**Code lines:** 20

**Contained by:** module `halo_mass_function_sheth_tormen`

**Modules used:** `cosmology_parameters critical_overdensity numerical_constants_math power_spectra`

**subroutine:** `halo_mass_function_sheth_tormen_initialize`

**Description:** Initializes the “Sheth-Tormen mass function” module.

**Code lines:** 9

**Contained by:** module `halo_mass_function_sheth_tormen`

**Modules used:** `iso_varying_string`

**file:** `structure_formation.halo_mass_function.Tinker2008.F90`

**Description:** Contains a module which generates a tabulated Tinker2008 halo mass function.

**Code lines:** 133

**module:** `halo_mass_function_tinker2008`

**Description:** Implements generation of a tabulated power-law primordial power spectrum.
19.1. Program units

Code lines: 113
Contained by: file structure_formation.halo_mass_function.Tinker2008.F90
Used by: subroutine halo_mass_function_initialize

function: halo_mass_function_differential_tinker2008
Description: Compute the Tinker et al. [2008] halo mass function.
Code lines: 58
Contained by: module halo_mass_function_tinker2008
Modules used: cosmology_functions cosmology_parameters
gsl linear_growth
numerical_interpolation power_spectra
virial_density_contrast

subroutine: halo_mass_function_tinker2008_initialize
Description: Initializes the “Tinker2008 mass function” module.
Code lines: 36
Contained by: module halo_mass_function_tinker2008
Modules used: fox_dom galacticus_error
galacticus_input_paths io_xml
iso_varying_string

file: structure_formation.halo_mass_function.modifier.simple_systematic.F90
Description: Contains a module which implements a simple systematic modifier of the dark matter halo mass function.
Code lines: 81

module: halo_mass_function_modifiers_simple_systematic
Description: Implements a simple systematic modifier of the dark matter halo mass function.
Code lines: 61
Contained by: file structure_formation.halo_mass_function.modifier.simple_systematic.F90
Used by: function halo_mass_function_differential

subroutine: halo_mass_function_modifier_simple_systematic
Description: Modify the halo mass function by applying a simple systematic shift in abundance.
Code lines: 42
Contained by: module halo_mass_function_modifiers_simple_systematic
Modules used: input_parameters

file: structure_formation.linear_growth.F90
Description: Contains a module which implements linear growth factor calculations.
Code lines: 339

module: linear_growth
Description: Implements the virial overdensity for halos.
Code lines: 319
Contained by: file structure_formation.linear_growth.F90
Modules used: gsl iso_varying_string
19. Source Code Documentation

(subroutine) interpolate_in_wavenumber
Description: Find interpolating factors in the wavenumber dimension for linear growth factor calculations.
Code lines: 19
Contained by: module linear_growth
Modules used: iso_c_binding numerical_interpolation

(function) linear_growth_factor
Description: Return the linear growth factor.
Code lines: 99
Contained by: module linear_growth
Modules used: cosmology_functions galacticus_error iso_c_binding numerical_interpolation

(function) linear_growth_factor_logarithmic_derivative
Description: Return the logarithmic derivative of the linear growth factor with respect to expansion factor, \( \frac{d \ln D}{d \ln a} \).
Code lines: 79
Contained by: module linear_growth
Modules used: cosmology_functions galacticus_error iso_c_binding numerical_interpolation

(subroutine) linear_growth_initialize
Description: Initializes the growth factor module.
Code lines: 52
Contained by: module linear_growth
Modules used: galacticus_error input_parameters linear_growth_simple

Used by:
- function prada2011concentration
- subroutine galacticus_growth_factor_output
- subroutine halo_mass_function_compute
- function intergalactic_medium_state_internal_odes
- subroutine critical_overdensity_kitayama_suto1996
- function power_spectrum_nonlinear_peacockdodds1996
- subroutine make_table
- program tests_linear_growth_eds
- program tests_linear_growth_dark_energy
- program tests_sphericalCollapse_dark_energy_omega_half
- program tests_sphericalCollapse_dark_energy_omega_two_thirds
- program tests_sphericalCollapse_dark_energy_lambda
- program tests_sphericalCollapse_dark_energy_open
- subroutine critical_overdensity_kitayama_suto1996
- function halo_mass_function_linear
- subroutine intergalactic_medium_state_internal_initialization
- function intergalactic_medium_state_internal_update
- subroutine make_table
- program tests_linear_growth_cosmological_constant
- program tests_linear_growth_open
19.1. Program units

**subroutine: linear_growth_state_retrieve**

*Description:* Reset the tabulation if state is to be retrieved. This will force tables to be rebuilt.

*Code lines:* 13

*Contained by:* module linear_growth

*Modules used:* memory_management

**file: structure_formation.linear_growth.simple.F90**

*Description:* Contains a module which implements calculations of linear growth factor in simple cosmologies. Ignores pressure terms for the growth of baryons and has no wavenumber dependence. Also assumes no growth of radiation perturbations.

*Code lines:* 194

**module: linear_growth_simple**

*Description:* Implements calculations of linear growth factor in simple cosmologies. Ignores pressure terms for the growth of baryons and has no wavenumber dependence. Also assumes no growth of radiation perturbations.

*Code lines:* 173

*Contained by:* file structure_formation.linear_growth.simple.F90

*Modules used:* cosmology_functions  cosmology_parameters
fgsl  iso_c_binding

*Used by:* subroutine galacticus_state_retrieve
subroutine galacticus_state_store
subroutine linear_growth_initialize

**subroutine: growth_factor_simple_initialize**

*Description:* Initializes the simple growth factor module.

*Code lines:* 11

*Contained by:* module linear_growth_simple

*Modules used:* iso_varying_string

**function: growthtableodes**

*Description:* System of differential equations to solve for the growth factor.

*Code lines:* 13

*Contained by:* module linear_growth_simple

**subroutine: linear_growth_factor_simple_tabulate**

*Description:* Returns the linear growth factor $D(a)$ for expansion factor $\text{aExp}ansion$, normalized such that $D(1) = 1$ for a simple matter plus cosmological constant cosmology.

*Code lines:* 85

*Contained by:* module linear_growth_simple

*Modules used:* memory_management  numerical_interpolation
numerical_ranges  ode_solver

**subroutine: linear_growth_simple_state_retrieve**

*Description:* Retrieve the tabulation state from the file.

*Code lines:* 9

*Contained by:* module linear_growth_simple

**subroutine: linear_growth_simple_state_store**
Description: Write the tabulation state to file.
Code lines: 8
Contained by: module linear_growth_simple

file: structure_formation.power_spectrum.F90
Description: Contains a module which implements the cosmological power spectrum.
Code lines: 290

module: power_spectra
Description: Implements the cosmological power spectrum.
Code lines: 270
Contained by: file structure_formation.power_spectrum.F90
Modules used: iso_varying_string
Used by: program tests_excursion_sets
function dark_matter_halo_mass_-accretion_time_zhao2009
function diemerkravtsov2014concentration
function nfw1996concentration
function nfw1996concentration
function dark_matter_profile_shape_-gao2008
subroutine halo_mass_function_compute
function intergalactic_medium_state_-internal_odes
module generalized_press_schechter_-branching
subroutine galacticusopen
subroutine merger_tree_write
function collapsing_mass_root
function excursion_sets_barrier_-gradient_critical_overdensity
function dark_matter_halo_bias_smt
function halo_mass_function_-differential_press_schechter
function halo_mass_function_-differential_tinker2008
function power_spectrum_nonlinear_-peacockdodds1996
program tests_power_spectrum
function expansion_factor_at_formations
function growthrateodes
function diemerkravtsov2014concentration
function prada2011concentration
function prada2011concentration
subroutine galacticus_linear_power_-spectrum_output
subroutine intergalactic_medium_state_-internal_initialize
function intergalactic_medium_state_-internal_update
module modified_press_schechter_-branching
subroutine galacticusopen
subroutine power_spectrum_compute
function excursion_sets_barrier_-critical_overdensity
function dark_matter_halo_bias_press_-schechter
function dark_matter_halo_bias_-tinker2010
function halo_mass_function_-tormen_differential
function power_spectrum_nonlinear_-cosmicemu
function power_spectrum_nonlinear_-linear
program tests_sigma

function: cosmological_mass_root_variance
Description: Computes the fractional mass fluctuation in real-space spherical top hats enclosing mass
Code lines: 13
Contained by: module power_spectra
19.1. Program units

function: cosmological_mass_root_variance_logarithmic_derivative
Description: Computes the logarithmic derivative in the fractional mass fluctuation in real-space spherical
top hats enclosing mass mass.
Code lines: 9
Contained by: module power_spectra

subroutine: cosmological_mass_root_variance_plus_logarithmic_derivative
Description: Returns both the fractional mass fluctuation in real-space spherical top hats enclosing mass
mass and its logarithmic derivative.
Code lines: 17
Contained by: module power_spectra

subroutine: initialize_cosmological_mass_variance
Description: Ensure that $\sigma(M)$ is tabulated over a range that includes logMass. The default normalization,
$\sigma_9 = 0.817$, is taken from Hinshaw et al. 2012; CMB+$H_0$+BAO).
Code lines: 84
Contained by: module power_spectra
Modules used: cosmological_mass_variance_filtered_power_spectrum
power_spectrum
galacticus_error
input_parameters
numerical_constants_math

function: mass_from_cosmolgical_root_variance
Description: Computes the mass corresponding to the given fractional mass fluctuation in real-space
spherical top hats.
Code lines: 32
Contained by: module power_spectra

function: power_spectrum
Description: Return the cosmological power spectrum for $k =$wavenumber [Mpc$^{-1}$].
Code lines: 35
Contained by: module power_spectra
Modules used: cosmology_parameters galacticus_error
numerical_constants_math

function: power_spectrum_dimensionless
Description: Return the dimensionless power spectrum, $\Delta^2(k)$, for $k =$wavenumber [Mpc$^{-1}$].
Code lines: 8
Contained by: module power_spectra
Modules used: numerical_constants_math

function: power_spectrum_logarithmic_derivative
Description: Return the logarithmic derivative of the power spectrum, $d \ln P(k)/d \ln k$, for $k =$wavenumber [Mpc$^{-1}$].
Code lines: 9
Contained by: module power_spectra
Modules used: primordial_power_spectra transfer_functions

function: sigma_8
19. Source Code Documentation

Description: Return the value of $\sigma_8$.
Code lines: 11
Contained by: module *power_spectra*

**file**: *structure_formation.power_spectrum.nonlinear.CosmicEmu.F90*
Description: Contains a module which implements the nonlinear power spectrum using the code of Lawrence et al. [2010].
Code lines: 163

**module**: *power_spectra_nonlinear_cosmicemu*
Description: Implements the nonlinear power spectrum using the code of Lawrence et al. [2010].
Code lines: 143
Contained by: file *structure_formation.power_spectrum.nonlinear.CosmicEmu.F90*
Modules used: *fgsl*
Used by: subroutine *power_spectrum_nonlinear_-
initialize*

**function**: *power_spectrum_nonlinear_cosmicemu*
Description: Return a nonlinear power spectrum equal using the code of Lawrence et al. [2010].
Code lines: 108
Contained by: module *power_spectra_nonlinear_cosmicemu*
Modules used: *cosmology_functions cosmology_parameters*
  *file_utilities fox_wxml*
  *galacticus_error galacticus_input_paths*
  *input_parameters iso_varying_string*
  *memory_management numerical_comparison*
  *numerical_interpolation power_spectra*
  *primordial_power_spectra system_command*

**subroutine**: *power_spectrum_nonlinear_cosmicemu_initialize*
Description: Initializes the “CosmicEmu” nonlinear power spectrum module.
Code lines: 10
Contained by: module *power_spectra_nonlinear_cosmicemu*
Modules used: *input_parameters iso_varying_string*

**file**: *structure_formation.power_spectrum.nonlinear.F90*
Description: Contains a module which implements the nonlinear power spectrum.
Code lines: 111

**module**: *power_spectra_nonlinear*
Description: Implements the nonlinear power spectrum.
Code lines: 91
Contained by: file *structure_formation.power_spectrum.nonlinear.F90*
Modules used: *iso_varying_string*
Used by: function *lss_integrand* subroutine *mass_function_covariance_-matrix*

**function**: *power_spectrum_nonlinear*
Description: Return the nonlinear power spectrum for $k = \text{wavenumber \ [Mpc}^{-1}]$ at the given cosmic $\text{time \ [Gyr]}$. 
19.1. Program units

subroutine: power_spectrum_nonlinear_initialize
Description: Initialize the nonlinear power spectrum module.
Code lines: 57
Contained by: module power_spectra_nonlinear
Modules used: galacticus_error input_parameters
power_spectra_nonlinear_cosmicemu power_spectra_nonlinear_linear
power_spectra_nonlinear_peacockdodds1996

Description: Contains a module which implements the nonlinear power spectrum using the algorithm of Peacock and Dodds [1996].
Code lines: 99

module: power_spectra_nonlinear_peacockdodds1996
Description: Implements the nonlinear power spectrum using the algorithm of Peacock and Dodds [1996].
Code lines: 79
Contained by: file structure_formation.power_spectrum.nonlinear.PeacockDodds1996.F90
Used by: subroutine power_spectrum_nonlinear_initialize

function: power_spectrum_nonlinear_peacockdodds1996
Description: Return a nonlinear power spectrum equal using the algorithm of Peacock and Dodds [1996].
Code lines: 55
Contained by: module power_spectra_nonlinear_peacockdodds1996
Modules used: cosmology_functions galacticus_error
linear_growth numerical_constants_math
power_spectra

subroutine: power_spectrum_nonlinear_peacockdodds1996_initialize
Description: Initializes the “Peacock-Dodds1996” nonlinear power spectrum module.
Code lines: 9
Contained by: module power_spectra_nonlinear_peacockdodds1996
Modules used: iso_varying_string

file: structure_formation.power_spectrum.nonlinear.linear.F90
Description: Contains a module which implements the nonlinear power spectrum as the linear power spectrum (useful mostly for testing).
Code lines: 53

module: power_spectra_nonlinear_linear
Description: Implements the nonlinear power spectrum as the linear power spectrum (useful mostly for testing).
Code lines: 33
Contained by: file structure_formation.power_spectrum.nonlinear.linear.F90
Used by: subroutine power_spectrum_nonlinear_initialize
function: power_spectrum_nonlinear_linear
Description: Return a nonlinear power spectrum equal to the linear power spectrum. (Useful mostly for testing.)
Code lines: 9
Contained by: module power_spectra_nonlinear_linear
Modules used: linear_growth power_spectra

subroutine: power_spectrum_nonlinear_linear_initialize
Description: Initializes the “linear” nonlinear power spectrum module.
Code lines: 9
Contained by: module power_spectra_nonlinear_linear
Modules used: iso_varying_string

file: structure_formation.power_spectrum.primordial.F90
Code lines: 185

module: primordial_power_spectra
Code lines: 167
Contained by: file structure_formation.power_spectrum.primordial.F90
Modules used: fgal iso_varying_string
Used by: subroutine galacticus_state_retrieve function power_spectrum_logarithmic_derivative
function power_spectrum_nonlinear_derivative function primordial_power_spectrum_transferred

subroutine: power_spectrum_initialize
Description: Initializes the transfer function module.
Code lines: 52
Contained by: module primordial_power_spectra
Modules used: galacticus_error input_parametersprimordial_power_spectrum_power_law

function: primordial_power_spectrum
Description: Return the CDM primordial power spectrum for \(k = \text{wavenumber} \, [\text{Mpc}^{-1}]\).
Code lines: 30
Contained by: module primordial_power_spectra
Modules used: numerical_interpolation

function: primordial_power_spectrum_logarithmic_derivative
Description: Return the logarithmic derivative CDM primordial power spectrum for \(k = \text{wavenumber} \, [\text{Mpc}^{-1}]\).
Code lines: 30
Contained by: module primordial_power_spectra
Modules used: numerical_interpolation

subroutine: primordial_power_spectrum_state_retrieve
Description: Reset the tabulation if state is to be retrieved. This will force tables to be rebuilt.
Code lines: 12
19.1. Program units

**Contained by:** module `primordial_power_spectra`
**Modules used:** `memory_management`

**file:** `structure_formation.power_spectrum.primordial.power_law.F90`
**Description:** Contains a module which generates a tabulated power-law primordial power spectrum.
**Code lines:** 151

**module:** `primordial_power_spectrum_power_law`
**Description:** Implements generation of a tabulated power-law primordial power spectrum. The default power spectrum parameters are taken from Hinshaw et al. 2012; CMB+$H_0$+BAO).
**Code lines:** 131
**Contained by:** file `structure_formation.power_spectrum.primordial.power_law.F90`
**Used by:** subroutine `galacticus_state_retrieve` subroutine `galacticus_state_store` subroutine `power_spectrum_initialize`

**subroutine:** `power_spectrum_power_law_tabulate`
**Description:** Tabulate a power-law primordial power spectrum.
**Code lines:** 34
**Contained by:** module `primordial_power_spectrum_power_law`
**Modules used:** `memory_management` `numerical_constants_math` `numerical_ranges`

**subroutine:** `primordial_power_spectrum_power_law_initialize`
**Description:** Initializes the “transfer function from CMBFast” module.
**Code lines:** 45
**Contained by:** module `primordial_power_spectrum_power_law`
**Modules used:** `input_parameters` `iso_varying_string`

**subroutine:** `primordial_power_spectrum_power_law_state_retrieve`
**Description:** Retrieve the tabulation state from the file.
**Code lines:** 10
**Contained by:** module `primordial_power_spectrum_power_law`
**Modules used:** `fgsl`

**subroutine:** `primordial_power_spectrum_power_law_state_store`
**Description:** Write the tabulation state to file.
**Code lines:** 9
**Contained by:** module `primordial_power_spectrum_power_law`
**Modules used:** `fgsl`

**file:** `structure_formation.power_spectrum.primordial.transferred.F90`
**Description:** Contains a module which implements the primordial power spectrum transferred to late times.
**Code lines:** 39

**module:** `primordial_power_spectra_transferred`
**Description:** Implements the primordial power spectrum transferred to late times.
**Code lines:** 19
**Contained by:** file `structure_formation.power_spectrum.primordial.transferred.F90`
19. Source Code Documentation

**Used by:**
- function `variance_integrand`
- function `variance_integrand_tophat`
- function `power_spectrum`

**function: primordial_power_spectrum_transferred**
- **Description:** Return the primordial power spectrum transferred to late times for \( k = \text{wavenumber} \ [\text{Mpc}^{-1}] \).
- **Code lines:** 9
- **Contained by:** module `primordial_power_spectra_transferred`
- **Modules used:** `primordial_power_spectra` `transfer_functions`

**file: structure_formation.power_spectrum.variance.window_function.F90**
- **Description:** Contains a module which implements window functions for computing the variance of the power spectrum.
- **Code lines:** 125

**module: power_spectrum_window_functions**
- **Description:** Implements window functions for computing the variance of the power spectrum.
- **Code lines:** 105
- **Contained by:** file `structure_formation.power_spectrum.variance.window_function.F90`
- **Modules used:** `iso_varying_string`
- **Used by:** function `variance_integral`
- **function variance_integrand**

**function: power_spectrum_window_function**
- **Description:** Returns the window function for power spectrum variance computation at the specified wavenumber (in Mpc\(^{-1}\)) for a given smoothingMass (in \( M_\odot \)).
- **Code lines:** 12
- **Contained by:** module `power_spectrum_window_functions`

**function: power_spectrum_window_function_wavenumber_maximum**
- **Description:** Returns the maximum wavenumber for which the window function for power spectrum variance computation is non-zero for a given smoothingMass (in \( M_\odot \)).
- **Code lines:** 12
- **Contained by:** module `power_spectrum_window_functions`

**subroutine: power_spectrum_window_functions_initialize**
- **Description:** Initialize the power spectrum window function module.
- **Code lines:** 56
- **Contained by:** module `power_spectrum_window_functions`
- **Modules used:** `galacticus_error` `input_parameters`
- `power_spectrum_window_functions_sharp_kspace` `power_spectrum_window_functions_th_kss_hybrid`
- `power_spectrum_window_functions_top_hat`

**file: structure_formation.power_spectrum.variance.window_function.sharp_kSpace.F90**
- **Description:** Contains a module which implements sharp in \( k \)-space window function for power spectrum variance computation.
- **Code lines:** 111

**module: power_spectrum_window_functions_sharp_kspace**
19.1. Program units

**Description:** Implements a sharp in k-space window function for power spectrum variance computation.

**Code lines:** 90

**Contained by:** file `structure_formation.power_spectrum.variance.window_function.sharp_kSpace.F90`

**Used by:** subroutine `power_spectrum_window_functions_initialize`

**Function:** `power_spectrum_window_function_sharp_kSpace`

**Description:** Top hat in real space window function Fourier transformed into k-space used in computing the variance of the power spectrum. The normalization of the filter is chosen such that, in real-space, \( W(r = 0) = 1 \). This results in a contained mass of \( M = 6\pi^2\bar{\rho}k_s^{-3} \) if \( k_s \) is the cut-off wavelength for the filter.

**Code lines:** 17

**Contained by:** module `power_spectrum_window_functions_sharp_kSpace`

**Function:** `power_spectrum_window_function_wavenumber_maximum_sharp_kSpace`

**Description:** Top hat in real space window function Fourier transformed into k-space used in computing the variance of the power spectrum. The normalization of the filter is chosen such that, in real-space, \( W(r = 0) = 1 \). This results in a contained mass of \( M = 6\pi^2\bar{\rho}k_s^{-3} \) if \( k_s \) is the cut-off wavelength for the filter.

**Code lines:** 9

**Contained by:** module `power_spectrum_window_functions_sharp_kSpace`

**Subroutine:** `power_spectrum_window_functions_sharp_kSpace_initialize`

**Description:** Initializes the “kSpaceSharp” power spectrum variance window function module.

**Code lines:** 44

**Contained by:** module `power_spectrum_window_functions_sharp_kSpace`

**Modules used:**
- `cosmology_parameters`
- `input_parameters`
- `iso_varying_string`
- `numerical_constants_math`

**File:** `structure_formation.power_spectrum.variance.window_function.top_hat.F90`

**Description:** Contains a module which implements top-hat window function for power spectrum variance computation.

**Code lines:** 86

**Module:** `power_spectrum_window_functions_top_hat`

**Description:** Implements top-hat window function for power spectrum variance computation.

**Code lines:** 65

**Contained by:** file `structure_formation.power_spectrum.variance.window_function.top_hat.F90`

**Used by:**
- function `variance_integrand_tophat`
- subroutine `power_spectrum_window_functions_initialize`

**Function:** `power_spectrum_window_function_top_hat`

**Description:** Top hat in real space window function Fourier transformed into k-space used in computing the variance of the power spectrum.

**Code lines:** 26

**Contained by:** module `power_spectrum_window_functions_top_hat`

**Modules used:**
- `cosmology_parameters`
- `numerical_constants_math`

**Function:** `power_spectrum_window_function_wavenumber_maximum_top_hat`
**Description:** Maximum wavenumber for a top hat in real space window function Fourier transformed into k-space used in computing the variance of the power spectrum.

**Code lines:** 9

**Contained by:** module `power_spectrum_window_functions_top_hat`

**Subroutine:** `power_spectrum_window_functions_top_hat_initialize`

**Description:** Initializes the “topHat” power spectrum variance window function module.

**Code lines:** 13

**Contained by:** module `power_spectrum_window_functions_top_hat`

**Modules used:** `iso_varying_string`

**File:** `structure_formation.power_spectrum.variance.window_function.top_hat_kspace_sharp_hybrid.F90`

**Description:** Contains a module which implements top-hat window function for power spectrum variance computation.

**Code lines:** 161

**Module:** `power_spectrum_window_functions_th_kss_hybrid`

**Description:** Implements top-hat window function for power spectrum variance computation.

**Code lines:** 140

**Contained by:** file `structure_formation.power_spectrum.variance.window_function.top_hat_kspace_sharp_hybrid.F90`

**Used by:** subroutine `power_spectrum_window_functions_initialize`

**Function:** `power_spectrum_window_function_th_kss_hybrid`

**Description:** Computes a window function for calculations of the variance in the power spectrum. Specifically, uses a convolution of top-hat real-space and sharp k-space window functions. The top-hat radius is \( r_{th} \), while the k-space cut-off wavenumber is \( k_s = a/r_s \), where \( a = \) [powerSpectrumWindowFunctionSharpKSpaceNormalization]. The two radii are chosen such that \( r_{th}^2 + r_s^2 = (3M/4\pi\rho)^{1/3} \) and \( r_s = \beta r_{th} \) where \( \beta = \) [powerSpectrumWindowFunctionSharpKSpaceTopHatRadiiRatio].

**Code lines:** 49

**Contained by:** module `power_spectrum_window_functions_th_kss_hybrid`

**Modules used:** `cosmology_parameters` `numerical_constants_math`

**Function:** `power_spectrum_window_function_wavenumber_maximum_th_kss_hybrid`

**Description:** Computes the maximum wavenumber at which the window function for calculations of the variance in the power spectrum is non-zero. Specifically, uses a convolution of top-hat real-space and sharp k-space window functions. The top-hat radius is \( r_{th} \), while the k-space cut-off wavenumber is \( k_s = a/r_s \), where \( a = \) [powerSpectrumWindowFunctionSharpKSpaceNormalization]. The two radii are chosen such that \( r_{th}^2 + r_s^2 = (3M/4\pi\rho)^{1/3} \) and \( r_s = \beta r_{th} \) where \( \beta = \) [powerSpectrumWindowFunctionSharpKSpaceTopHatRadiiRatio].

**Code lines:** 13

**Contained by:** module `power_spectrum_window_functions_th_kss_hybrid`

**Subroutine:** `power_spectrum_window_functions_th_kss_hybrid_initialize`

**Description:** Initializes the “topHatKSpaceSharpHybrid” power spectrum variance window function module.

**Code lines:** 55
19.1. Program units

**modules**: module power_spectrum_window_functions_th_kss_hybrid

**modules used**:
- cosmology_parameters
- input_parameters
- iso_varying_string
- numerical_constants_math

**file**: structure_formation.spherical_collapse.matter_dark_energy.F90

**Description**: Contains a module which implements calculations of spherical top hat collapse in cosmologies containing matter and dark energy.

**Code lines**: 351

**modules**: module spherical_collapse_matter_dark_energy

**Description**: Implements calculations of spherical top hat collapse in cosmologies containing matter and dark energy.

**Code lines**: 331

**Contained by**:
- file structure_formation.spherical_collapse.matter_dark_energy.F90

**Modules used**:
- cosmology_functions
- fgsl
- iso_c_binding
- iso_varying_string

**Used by**:
- subroutine galacticus_state_retrieve
- subroutine galacticus_state_store
- subroutine critical_overdensity_initialize
- subroutine sphericalcollapsematterderetabulate

**function**: expansionrateperturbation

**Description**: Return the expansion rate of a spherical top-hat perturbation in a dark energy universe given an initial perturbation amplitude epsilonPerturbation.

**Code lines**: 8

**Contained by**:
- module spherical_collapse_matter_dark_energy

**subroutine**: make_table

**Description**: Tabulate $\delta_{\text{crit}}$ or $\Delta_{\text{vir}}$ vs. time.

**Code lines**: 140

**Contained by**:
- module spherical_collapse_matter_dark_energy

**Modules used**:
- galacticus_display
- galacticus_error
- input_parameters
- linear_growth
- root_finder
- tables

**subroutine**: perturbation_dynamics_solver

**Description**: Integrate the dynamics of a spherical top-hat perturbation in a dark energy universe given an initial perturbation amplitude epsilonPerturbation.

**Code lines**: 45

**Contained by**:
- module spherical_collapse_matter_dark_energy

**Modules used**:
- fodeiv2
- odeiv2_solver

**function**: perturbationnodes

**Code lines**: 19

**Contained by**:
- module spherical_collapse_matter_dark_energy

**function**: radiusperturbation

**Description**: Return the radius of a spherical top-hat perturbation in a dark energy universe given an initial perturbation amplitude epsilonPerturbation.

**Code lines**: 1877
19. Source Code Documentation

**Code lines:** 8  
**Contained by:** module `spherical_collapse_matter_dark_energy`

**subroutine:** `spherical_collapse_dark_energy_critical_overdensity`  
**Description:** Tabulate the critical overdensity for collapse for the spherical collapse model.  
**Code lines:** 12  
**Contained by:** module `spherical_collapse_matter_dark_energy`  
**Modules used:** `tables`

**subroutine:** `spherical_collapse_dark_energy_delta_critical_initialize`  
**Description:** Initializes the $\delta_{\text{crit}}$ calculation for the spherical collapse module.  
**Code lines:** 8  
**Contained by:** module `spherical_collapse_matter_dark_energy`

**subroutine:** `spherical_collapse_dark_energy_virial_density_contrast_tabulate`  
**Description:** Tabulate the virial density contrast for the spherical collapse model.  
**Code lines:** 11  
**Contained by:** module `spherical_collapse_matter_dark_energy`  
**Modules used:** `tables`

**subroutine:** `spherical_collapse_matter_dark_energy_state_retrieve`  
**Description:** Retrieve the tabulation state from the file.  
**Code lines:** 8  
**Contained by:** module `spherical_collapse_matter_dark_energy`

**subroutine:** `spherical_collapse_matter_dark_energy_state_store`  
**Description:** Write the tabulation state to file.  
**Code lines:** 8  
**Contained by:** module `spherical_collapse_matter_dark_energy`

**file:** `structure_formation.spherical_collapse.matter_lambda.F90`  
**Description:** Contains a module which implements calculations of spherical top hat collapse in cosmologies containing matter and a cosmological constant.  
**Code lines:** 311

**module:** `spherical_collapse_matter_lambda`  
**Code lines:** 290  
**Contained by:** file `structure_formation.spherical_collapse.matter_lambda.F90`  
**Modules used:** `fgsl`  
**Used by:** subroutine `galacticus_state_retrieve`  
**Used by:** subroutine `critical_overdensity_initialize`  
**Used by:** subroutine `spherical_collapse_matter_lambda_tabulate`

**function:** `amaximumroot`  
**Description:** Root function for maximum expansion radius.  
**Code lines:** 6  
**Contained by:** module `spherical_collapse_matter_lambda`
function: collapseroor
  Code lines:  6
  Contained by: module sphericalCollapseMatterLambda

subroutine: make_table
  Description: Tabulate $\delta_{\text{crit}}$ or $\Delta_{\text{vir}}$ vs. time.
  Code lines:  100
  Contained by: module sphericalCollapseMatterLambda
  Modules used: cosmology_functions, galacticus_error, kind_numbers, linear_growth, root_finder, tables

function: perturbation_integrand
  Code lines:  15
  Contained by: module sphericalCollapseMatterLambda

function: perturbation_maximum_radius
  Description: Find the maximum radius of a perturbation with initial curvature $\epsilon_{\text{Perturbation}}$.
  Code lines:  37
  Contained by: module sphericalCollapseMatterLambda
  Modules used: root_finder

subroutine: sphericalCollapseDeltaCriticalInitialize
  Description: Initializes the $\delta_{\text{crit}}$ calculation for the spherical collapse module.
  Code lines:  9
  Contained by: module sphericalCollapseMatterLambda
  Modules used: iso_varuing_string

subroutine: sphericalCollapseMatterLambdaDeltaVirialTabulate
  Description: Tabulate the virial density contrast for the spherical collapse model.
  Code lines:  11
  Contained by: module sphericalCollapseMatterLambda
  Modules used: tables

subroutine: sphericalCollapseCriticalOverdensity
  Description: Tabulate the critical overdensity for collapse for the spherical collapse model.
  Code lines:  12
  Contained by: module sphericalCollapseMatterLambda
  Modules used: tables

subroutine: sphericalCollapseMatterLambdaStateRetrieve
  Description: Retrieve the tabulation state from the file.
  Code lines:  8
  Contained by: module sphericalCollapseMatterLambda

subroutine: sphericalCollapseMatterLambdaStateStore
  Description: Write the tabulation state to file.
  Code lines:  8
function: tcollapse
Code lines: 28
Contained by: module spherical_collapse_matter_lambda
Modules used: numerical_integration

file: structureFormation.transfer_function.BBKS.F90
Description: Contains a module which generates a tabulated transfer function using the BBKS fitting formula.
Code lines: 148

module: transfer_function_bbks
Description: Implements generation of a tabulated transfer function using the BBKS fitting formula.
Code lines: 128
Contained by: file structureFormation.transfer_function.BBKS.F90
Modules used: iso_varying_string
Used by: subroutine galacticus_state_retrieve subroutine galacticus_state_store subroutine transfer_function_initialize

subroutine: transfer_function_bbks_initialize
Description: Initializes the “transfer function from BBKS” module.
Code lines: 27
Contained by: module transfer_function_bbks
Modules used: input_parameters

subroutine: transfer_function_bbks_make
Description: Build a transfer function using the BBKS fitting formula.
Code lines: 38
Contained by: module transfer_function_bbks
Modules used: cosmology_parameters memory_management numerical_constants_math numerical_ranges

subroutine: transfer_function_bbks_state_retrieve
Description: Retrieve the tabulation state from the file.
Code lines: 10
Contained by: module transfer_function_bbks
Modules used: fgsl

subroutine: transfer_function_bbks_state_store
Description: Write the tabulation state to file.
Code lines: 9
Contained by: module transfer_function_bbks
Modules used: fgsl

function: transfer_function_half_mode_mass_null
Description: Compute the mass corresponding to the wavenumber at which the transfer function is suppressed by a factor of two relative to a CDM transfer function. Not supported in this implementation.
Code lines: 8
Contained by: module transfer_function_bbks
19.1. Program units

Modules used: galacticus_error

file: structure_formation.transfer_function.CAMB.F90
Description: Contains a module which generates a tabulated transfer function using CAMB.
Code lines: 131

module: transfer_function_camb
Description: Implements generation of a tabulated transfer function using CAMB.
Code lines: 111
Contained by: file structure_formation.transfer_function.CAMB.F90
Modules used: iso_varying_string
Used by: subroutine transfer_function_initialize

subroutine: transfer_function_camb_initialize
Code lines: 11
Contained by: module transfer_function_camb

subroutine: transfer_function_camb_make
Description: Build a transfer function using CAMB.
Code lines: 65
Contained by: module transfer_function_camb
Modules used: cosmology_parameters fox_wxml
galacticus_input_paths input_parameters
numerical_constants_astronomical string_handling
system_command transfer_functions_file

function: transfer_function_half_mode_mass_null
Description: Compute the mass corresponding to the wavenumber at which the transfer function is suppressed by a factor of two relative to a CDM transfer function. Not supported in this implementation.
Code lines: 8
Contained by: module transfer_function_camb
Modules used: galacticus_error

file: structure_formation.transfer_function.Eisenstein_Hu.F90
Description: Contains a module which generates a tabulated transfer function using the Eisenstein & Hu fitting formula.
Code lines: 267

module: transfer_function_eisenstein_hu
Description: Implements generation of a tabulated transfer function using the Eisenstein & Hu fitting formula.
Code lines: 245
Contained by: file structure_formation.transfer_function.Eisenstein_Hu.F90
Modules used: iso_varying_string
Used by: subroutine galacticus_state_retrieve subroutine galacticus_state_store subroutine transfer_function_initialize

subroutine: transfer_function_eisenstein_hu_initialize
Description: Initializes the “transfer function from Eisenstein & Hu” module.
19. Source Code Documentation

Code lines: 79
Contained by: module transfer_function_eisenstein_hu
Modules used: input_parameters

**subroutine: transfer_function_eisenstein_hu_make**

*Description:* Build a transfer function using the Eisenstein and Hu [1999] fitting formula. Includes a modification for warm dark matter using the fitting function of Bode et al. (2001; as re-expressed by Barkana et al. 2001) to impose a cut-off below a specified \( \text{transferFunctionWdmCutOffScale} \).

Code lines: 88
Contained by: module transfer_function_eisenstein_hu
Modules used: cosmology_parameters memory_management numerical_constants_math numerical_ranges

subroutine: transfer_function_eisenstein_hu_state_retrieve

*Description:* Retrieve the tabulation state from the file.

Code lines: 10
Contained by: module transfer_function_eisenstein_hu

subroutine: transfer_function_eisenstein_hu_state_store

*Description:* Write the tablulation state to file.

Code lines: 9
Contained by: module transfer_function_eisenstein_hu

function: transfer_function_half_mode_mass_eisenstein_hu

*Description:* Find the half-mode mass for WDM calculations using the result from Schneider et al. (2012; their eqns. 8 & 9).

Code lines: 17
Contained by: module transfer_function_eisenstein_hu
Modules used: cosmology_parameters numerical_constants_math

file: structure_formation.transfer_function.F90

Code lines: 209

module: transfer_functions

Code lines: 191
Contained by: file structure_formation.transfer_function.F90
Modules used: fgs1 iso_varying_string
Used by: function wdmconcentration function wdmconcentration
subroutine galacticus_state_retrieve function power_spectrum_logarithmic_derivative

function: transfer_function

*Description:* Return the transfer function for \( k = \text{wavenumber} \) [Mpc\(^{-1}\)].

Code lines: 30
19.1. Program units

**Contained by:** module transfer_functions  
**Modules used:** numerical_interpolation

**function:** transfer_function_half_mode_mass  
**Description:** Return the mass (in $M_\odot$) corresponding to the wavenumber at which the transfer function is suppressed by a factor of two due to small-scale dark matter particle physics.  
**Code lines:** 13  
**Contained by:** module transfer_functions

**subroutine:** transfer_function_initialize  
**Description:** Initializes the transfer function module.  
**Code lines:** 60  
**Contained by:** module transfer_functions  
**Modules used:** galacticus_error input_parameters  
transfer_function_bbks transfer_function_camb  
transfer_function_eisenstein_hu transfer_function_null  
transfer_functions_file

**function:** transfer_function_logarithmic_derivative  
**Description:** Return the logarithmic derivative of the transfer function for $k =$wavenumber $[\text{Mpc}^{-1}]$.  
**Code lines:** 30  
**Contained by:** module transfer_functions  
**Modules used:** numerical_interpolation

**subroutine:** transfer_function_state_retrieve  
**Description:** Reset the tabulation if state is to be retrieved. This will force tables to be rebuilt.  
**Code lines:** 12  
**Contained by:** module transfer_functions  
**Modules used:** memory_management

**interface:** transfer_function_tabulate_template  
**Code lines:** 6  
**Contained by:** module transfer_functions

**subroutine:** transfer_function_tabulate_template  
**Code lines:** 4  
**Contained by:** interface transfer_function_tabulate_template

**file:** structure_formation.transfer_function.file.F90  
**Description:** Contains a module which reads a tabulated transfer function from a file.  
**Code lines:** 261

**module:** transfer_functions_file  
**Description:** Implements reading of a tabulated transfer function from a file.  
**Code lines:** 241  
**Contained by:** file structure_formation.transfer_function.file.F90  
**Modules used:** iso_varying_string  
**Used by:** subroutine transfer_function_camb_make subroutine transfer_function_initialize
subroutine: transfer_function_file_initialize
Description: Initializes the “transfer function from file” module.
Code lines: 23
Contained by: module transfer_functions_file
Modules used: input_parameters

subroutine: transfer_function_file_read
Description: Reads a transfer function from an XML file.
Code lines: 151
Contained by: module transfer_functions_file
Modules used: cosmology_parameters fox_dom
galacticus_display galacticus_error
io_xml memory_management
numerical_comparison numerical_constants_math
numerical_ranges

function: transfer_function_half_mode_mass_null
Description: Compute the mass corresponding to the wavenumber at which the transfer function is suppressed by a factor of two relative to a CDM transfer function. Not supported in this implementation.
Code lines: 8
Contained by: module transfer_functions_file
Modules used: galacticus_error

function: transfer_function_named_file_format_version
Description: Return the current file format version of transfer function files.
Code lines: 6
Contained by: module transfer_functions_file

subroutine: transfer_function_named_file_read
Description: Read the transfer function from a file specified by fileName.
Code lines: 15
Contained by: module transfer_functions_file

file: structure_formation.transfer_function.null.F90
Description: Contains a module which generates a null transfer function.
Code lines: 88

module: transfer_function_null
Description: Implements a null transfer function.
Code lines: 68
Contained by: file structure_formation.transfer_function.null.F90
Used by: subroutine transfer_function_initialize

function: transfer_function_half_mode_mass_null
Description: Compute the mass corresponding to the wavenumber at which the transfer function is suppressed by a factor of two relative to a CDM transfer function. Not supported in this implementation.
Code lines: 8
Contained by: module transfer_function_null
19.1. Program units

Modules used: galacticus_error

subroutine: transfer_function_null_initialize
Description: Initializes the “null transfer function” module.
Code lines: 13
Contained by: module transfer_function_null
Modules used: iso_varying_string

subroutine: transfer_function_null_make
Description: Build a null transfer function.
Code lines: 25
Contained by: module transfer_function_null
Modules used: memory_management numerical_constants_math numerical_ranges

file: structure_formation.virial_density_contrast.Bryan_Norman.F90
Code lines: 107

function: bryannorman1998defaultconstructor
Description: Default constructor for the bryanNorman1998 dark matter halo virial density contrast class.
Code lines: 20
Contained by: file structure_formation.virial_density_contrast.Bryan_Norman.F90
Modules used: cosmology_parameters galacticus_error numerical_comparison

function: bryannorman1998densitycontrast
Description: Return the virial density contrast at the given epoch, assuming the fitting function of Bryan and Norman [1998].
Code lines: 20
Contained by: file structure_formation.virial_density_contrast.Bryan_Norman.F90
Modules used: cosmology_functions numerical_constants_math

function: bryannorman1998densitycontrastrateofchange
Description: Return the virial density contrast at the given epoch, assuming the fitting function of Bryan and Norman [1998].
Code lines: 20
Contained by: file structure_formation.virial_density_contrast.Bryan_Norman.F90
Modules used: cosmology_functions numerical_constants_math

interface: virialdensitycontrastbryannorman1998
Description: Constructors for the bryanNorman1998 dark matter halo virial density contrast class.
Code lines: 3
Contained by: file structure_formation.virial_density_contrast.Bryan_Norman.F90

file: structure_formation.virial_density_contrast.F90
Description: Contains a module which provides a class implementing the virial density contrast for halos.
Code lines: 1093
module: virial_density_contrast


Code lines: 1073

Contained by: file structure_formation.virial_density_contrast.F90

Modules used: fgal

module dark_matter_halo_scales

module dark_matter_profiles_concentration

function nfw1996concentration

function dark_matter_halos_mass_loss_rate_vandenbosch

module dark_matter_halo_scales

module dark_matter_profiles_concentration

function nfw1996concentration

function dark_matter_profile_scale

subroutine galacticus_state_retrieve

subroutine halomass_function_compute

function halo_mass_function_differential_tinker2008

program tests_sphericalCollapse__dark_energy_lamda

function: bryannorman1998defaultconstructor

Description: Default constructor for the bryanNorman1998 dark matter halo virial density contrast class.

Code lines: 20

Contained by: module virial_density_contrast

Modules used: cosmology_parameters galacticus_error numerical_comparison

function: bryannorman1998densitycontrast

Description: Return the virial density contrast at the given epoch, assuming the fitting function of Bryan and Norman [1998].

Code lines: 20

Contained by: module virial_density_contrast

Modules used: cosmology_functions numerical_constants_math

function: bryannorman1998densitycontratasteofchange

Description: Return the virial density contrast at the given epoch, assuming the fitting function of Bryan and Norman [1998].
19.1. Program units

- **fixedconstructor**
  - Description: Constructor for the fixed dark matter halo virial density contrast class.
  - Code lines: 12
  - Contained by: module `virial_density_contrast`
  - Modules used: `cosmology_functions`, `numerical_constants_math`

- **fixeddefaultconstructor**
  - Description: Default constructor for the fixed dark matter halo virial density contrast class.
  - Code lines: 47
  - Contained by: module `virial_density_contrast`
  - Modules used: `galacticus_error`

- **fixeddensitycontrast**
  - Description: Return the virial density contrast at the given epoch, assuming a fixed contrast.
  - Code lines: 17
  - Contained by: module `virial_density_contrast`
  - Modules used: `cosmology_functions`

- **fixeddensitycontrastrateofchange**
  - Description: Return the virial density contrast at the given epoch, assuming a fixed contrast.
  - Code lines: 20
  - Contained by: module `virial_density_contrast`
  - Modules used: `cosmology_functions`

- **friendsoffriendsconstructor**
  - Description: Generic constructor for the friendsOfFriends dark matter halo virial density contrast class.
  - Code lines: 10
  - Contained by: module `virial_density_contrast`
  - Modules used: `input_parameters`

- **friendsoffriendsdefaultconstructor**
  - Description: Default constructor for the friendsOfFriends dark matter halo virial density contrast class.
  - Code lines: 39
  - Contained by: module `virial_density_contrast`
  - Modules used: `input_parameters`

- **friendsoffriendsdensitycontrast**
  - Description: Return the virial density contrast at the given epoch, based on the friends-of-friends algorithm linking length.
  - Code lines: 12
  - Contained by: module `virial_density_contrast`
  - Modules used: `numerical_constants_math`

- **friendsoffriendsdensitycontrastrateofchange**
  - Description: Return the virial density contrast at the given epoch, based on the friends-of-friends algorithm linking length.
  - Code lines: 10
function: kitayamasuto1996defaultconstructor
Description: Default constructor for the kitayamaSuto1996 dark matter halo virial density contrast class.
Code lines: 6
Contained by: module virial_density_contrast
Modules used: numerical_constants_math

function: kitayamasuto1996densitycontrast
Description: Return the virial density contrast at the given epoch, assuming the fitting function of Kitayama and Suto [1996].
Code lines: 15
Contained by: module virial_density_contrast
Modules used: cosmology_functions numerical_constants_math

function: kitayamasuto1996densitycontrastrateofchange
Description: Return the virial density contrast at the given epoch, assuming the fitting function of Kitayama and Suto [1996].
Code lines: 15
Contained by: module virial_density_contrast
Modules used: cosmology_functions numerical_constants_math

function: sphericalcollapsematterdefaultconstructor
Description: Default constructor for the sphericalCollapseMatterDE dark matter halo virial density contrast class.
Code lines: 8
Contained by: module virial_density_contrast
Modules used: input_parameters

subroutine: sphericalcollapsematterderetabulate
Description: Recompute the look-up tables for virial density contrast.
Code lines: 21
Contained by: module virial_density_contrast
Modules used: spherical_collapse_matter_dark_energy

function: sphericalcollapsematterlambdadefaultconstructor
Description: Default constructor for the sphericalCollapseMatterLambda dark matter halo virial density contrast class.
Code lines: 8
Contained by: module virial_density_contrast
Modules used: input_parameters

function: sphericalcollapsematterlambdadensitycontrast
Description: Return the virial density contrast at the given epoch, based spherical collapse in a matter plus cosmological constant universe.
Code lines: 38
Contained by: module virial_density_contrast
Modules used: cosmology_functions galacticus_error

function: sphericalcollapsematterlambdcontrastrateofchange
19.1. Program units

**Description:** Return the virial density contrast at the given epoch, based spherical collapse in a matter plus cosmological constant universe.

**Code lines:** 38

**Contained by:** module `virial_density_contrast`

**Modules used:** `cosmology_functions` `galacticus_error`

**subroutine:** `sphericalcollapsematterlambdadestructor`

**Description:** Destructor for the `sphericalCollapseMatterLambda` dark matter halo virial density contrast class.

**Code lines:** 10

**Contained by:** module `virial_density_contrast`

**subroutine:** `sphericalcollapsematterlambdaretabulate`

**Description:** Recompute the look-up tables for virial density contrast.

**Code lines:** 21

**Contained by:** module `virial_density_contrast`

**Modules used:** `spherical_collapse_matter_lambda`

**subroutine:** `sphericalcollapsematterlambdastaterestore`

**Description:** Reset the tabulation if state is to be retrieved. This will force tables to be rebuilt.

**Code lines:** 11

**Contained by:** module `virial_density_contrast`

**Modules used:** `fgsl`

**subroutine:** `sphericalcollapsematterlambdastatestore`

**Description:** Reset the tabulation if state is to be retrieved. This will force tables to be rebuilt.

**Code lines:** 10

**Contained by:** module `virial_density_contrast`

**Modules used:** `fgsl`

**interface:** `virialdensitycontrast`

**Code lines:** 3

**Contained by:** module `virial_density_contrast`

**interface:** `virialdensitycontrastbryannorman1998`

**Description:** Constructors for the `bryanNorman1998` dark matter halo virial density contrast class.

**Code lines:** 3

**Contained by:** module `virial_density_contrast`

**type:** `virialdensitycontrastclass`

**Code lines:** 42

**Contained by:** module `virial_density_contrast`

**function:** `virialdensitycontrastconstructordefault`

**Description:** Return a pointer to the default `virialDensityContrast` object.

**Code lines:** 8

**Contained by:** module `virial_density_contrast`

**function:** `virialdensitycontrastconstructornamed`
19. Source Code Documentation

**Description:** Return a pointer to a newly created `virialDensityContrast` object of the specified type.

**Code lines:** 58

**Contained by:** module `virial_density_contrast`

**Modules used:** `galacticus_error` `iso_varying_string`

**function:** virialdensitycontrastdensitycontrastnull

**Description:** Returns the virial density contrast at the given epoch.

**Code lines:** 9

**Contained by:** module `virial_density_contrast`

**Modules used:** `galacticus_error`

**function:** virialdensitycontrastdensitycontrastrateofchangenull

**Description:** Returns the rate of change of virial density contrast at the given epoch.

**Code lines:** 9

**Contained by:** module `virial_density_contrast`

**Modules used:** `galacticus_error`

**subroutine:** virialdensitycontrastdostateretrieve

**Description:** Retrieve the state from file.

**Code lines:** 10

**Contained by:** module `virial_density_contrast`

**subroutine:** virialdensitycontrastdostatestore

**Description:** Store the state to file.

**Code lines:** 10

**Contained by:** module `virial_density_contrast`

**interface:** virialdensitycontrastfixed

**Description:** Constructors for the fixed dark matter halo virial density contrast class.

**Code lines:** 4

**Contained by:** module `virial_density_contrast`

**interface:** virialdensitycontrastfriendsoffriends

**Description:** Constructors for the friendsOfFriends dark matter halo virial density contrast class.

**Code lines:** 4

**Contained by:** module `virial_density_contrast`

**subroutine:** virialdensitycontrastinitialize

**Description:** Initialize the default `virialDensityContrast` object.

**Code lines:** 75

**Contained by:** module `virial_density_contrast`

**Modules used:** `galacticus_error` `input_parameters` `iso_varying_string`

**function:** virialdensitycontrastisfinalizable

**Description:** Return true if this object can be finalized.

**Code lines:** 7

**Contained by:** module `virial_density_contrast`
interface: virialdensitycontrastkitayamasuto1996
Description:  Constructors for the kitayamaSuto1996 dark matter halo virial density contrast class.
Code lines: 3
Contained by: module virial_density_contrast

interface: virialdensitycontrastsphericalcollapsematterde
Description:  Constructors for the sphericalCollapseMatterDE dark matter halo virial density contrast class.
Code lines: 3
Contained by: module virial_density_contrast

interface: virialdensitycontrastsphericalcollapsematterlambda
Description:  Constructors for the sphericalCollapseMatterLambda dark matter halo virial density contrast class.
Code lines: 3
Contained by: module virial_density_contrast

subroutine: virialdensitycontraststaterestore
Description:  Restore the state of the object to file.
Code lines: 9
Contained by: module virial_density_contrast
Modules used: fgsl

subroutine: virialdensitycontraststatestore
Description:  Store the state of the object to file.
Code lines: 9
Contained by: module virial_density_contrast
Modules used: fgsl

Description:  An implementation of Kitayama and Suto [1996] dark matter halo virial density contrasts.
Code lines: 79

function: kitayamasuto1996defaultconstructor
Description:  Default constructor for the kitayamaSuto1996 dark matter halo virial density contrast class.
Code lines: 6

function: kitayamasuto1996densitycontrast
Description:  Return the virial density contrast at the given epoch, assuming the fitting function of Kitayama and Suto [1996].
Code lines: 15
Modules used: cosmology_functions
numerical_constants_math

function: kitayamasuto1996densitycontrastrateofchange
Description:  Return the virial density contrast at the given epoch, assuming the fitting function of Kitayama and Suto [1996].
Code lines: 15
19. Source Code Documentation

**Contained by:** file `structure_formation.virial_density_contrast.Kitayama_Suto1996.F90`  
**Modules used:** `cosmology_functions` `numerical_constants_math`

**interface:** `virialdensitycontrastkitayamasuto1996`  
**Description:** Constructors for the `kitayamaSuto1996` dark matter halo virial density contrast class.  
**Code lines:** 3  
**Contained by:** file `structure_formation.virial_density_contrast.Kitayama_Suto1996.F90`

**file:** `structure_formation.virial_density_contrast.fixed.F90`  
**Description:** An implementation of fixed dark matter halo virial density contrasts.  
**Code lines:** 156

**function:** `fixedconstructor`  
**Description:** Constructor for the `fixed` dark matter halo virial density contrast class.  
**Code lines:** 12  
**Contained by:** file `structure_formation.virial_density_contrast.fixed.F90`  
**Modules used:** `galacticus_error`

**function:** `fixeddefaultconstructor`  
**Description:** Default constructor for the `fixed` dark matter halo virial density contrast class.  
**Code lines:** 47  
**Contained by:** file `structure_formation.virial_density_contrast.fixed.F90`  
**Modules used:** `galacticus_error` `input_parameters`

**function:** `fixeddensitycontrast`  
**Description:** Return the virial density contrast at the given epoch, assuming a fixed contrast.  
**Code lines:** 17  
**Contained by:** file `structure_formation.virial_density_contrast.fixed.F90`  
**Modules used:** `cosmology_functions`

**function:** `fixeddensitycontrastrateofchange`  
**Description:** Return the virial density contrast at the given epoch, assuming a fixed contrast.  
**Code lines:** 20  
**Contained by:** file `structure_formation.virial_density_contrast.fixed.F90`  
**Modules used:** `cosmology_functions`

**interface:** `virialdensitycontrastfixed`  
**Description:** Constructors for the `fixed` dark matter halo virial density contrast class.  
**Code lines:** 4  
**Contained by:** file `structure_formation.virial_density_contrast.fixed.F90`

**file:** `structure_formation.virial_density_contrast.friends-of-friends.F90`  
**Description:** An implementation of dark matter halo virial density contrasts based on a friends-of-friends linking length.  
**Code lines:** 124

**function:** `friendsoffriendsconstructor`  
**Description:** Generic constructor for the `friendsOfFriends` dark matter halo virial density contrast class.  
**Code lines:** 10
19.1. Program units

**Contained by:** file `structure_formation.virial_density_contrast.friends-of-friends.F90`
**Modules used:** `input_parameters`

**function:** `friendsoffriendsdefaultconstructor`
**Description:** Default constructor for the `friendsOfFriends` dark matter halo virial density contrast class.
**Code lines:** 39
**Contained by:** file `structure_formation.virial_density_contrast.friends-of-friends.F90`
**Modules used:** `input_parameters`

**function:** `friendsoffriendsdensitycontrast`
**Description:** Return the virial density contrast at the given epoch, based on the friends-of-friends algorithm linking length.
**Code lines:** 12
**Contained by:** file `structure_formation.virial_density_contrast.friends-of-friends.F90`
**Modules used:** `numerical_constants_math`

**function:** `friendsoffriendsdensitycontrastrateofchange`
**Description:** Return the virial density contrast at the given epoch, based on the friends-of-friends algorithm linking length.
**Code lines:** 10
**Contained by:** file `structure_formation.virial_density_contrast.friends-of-friends.F90`
**Modules used:** `numerical_constants_math`

**interface:** `virialdensitycontrastfriendsoffriends`
**Description:** Constructors for the `friendsOfFriends` dark matter halo virial density contrast class.
**Code lines:** 4
**Contained by:** file `structure_formation.virial_density_contrast.friends-of-friends.F90`

**file:** `structure_formation.virial_density_contrast.spherical_collapse_matter_dark_energy.F90`
**Description:** An implementation of dark matter halo virial density contrasts based on spherical collapse in a matter plus dark energy universe.
**Code lines:** 70
**Modules used:** `tables`

**function:** `sphericalcollapsematterdedefaultconstructor`
**Description:** Default constructor for the `sphericalCollapseMatterDE` dark matter halo virial density contrast class.
**Code lines:** 8
**Contained by:** file `structure_formation.virial_density_contrast.spherical_collapse_matter_dark_energy.F90`
**Modules used:** `input_parameters`

**subroutine:** `sphericalcollapsematterderetabulate`
**Description:** Recompute the look-up tables for virial density contrast.
**Code lines:** 21
**Contained by:** file `structure_formation.virial_density_contrast.spherical_collapse_matter_dark_energy.F90`
**Modules used:** `sphericalCollapseMatterDE`

**interface:** `virialdensitycontrastsphericalcollapsematterde`
**Description:** Constructors for the `sphericalCollapseMatterDE` dark matter halo virial density contrast class.
**Code lines:** 3

1893
19. Source Code Documentation

Contained by: file `structure_formation.virial_density_contrast.sphericalCollapseMatterLambda.F90`

Description: An implementation of dark matter halo virial density contrasts based on spherical collapse in a matter plus cosmological constant universe.

Code lines: 204

Modules used: `tables`

Function: `sphericalCollapseMatterLambdaDefaultConstructor`

Description: Default constructor for the `sphericalCollapseMatterLambda` dark matter halo virial density contrast class.

Code lines: 8

Contained by: file `structure_formation.virial_density_contrast.sphericalCollapseMatterLambda.F90`

Modules used: `input_parameters`

Function: `sphericalCollapseMatterLambdaDensityContrast`

Description: Return the virial density contrast at the given epoch, based spherical collapse in a matter plus cosmological constant universe.

Code lines: 38

Contained by: file `structure_formation.virial_density_contrast.sphericalCollapseMatterLambda.F90`

Modules used: `cosmology_functions` `galacticus_error`

Function: `sphericalCollapseMatterLambdaDensityContrastRateOfChange`

Description: Return the virial density contrast at the given epoch, based spherical collapse in a matter plus cosmological constant universe.

Code lines: 38

Contained by: file `structure_formation.virial_density_contrast.sphericalCollapseMatterLambda.F90`

Modules used: `cosmology_functions` `galacticus_error`

Subroutine: `sphericalCollapseMatterLambdaDestructor`

Description: Destructor for the `sphericalCollapseMatterLambda` dark matter halo virial density contrast class.

Code lines: 10

Contained by: file `structure_formation.virial_density_contrast.sphericalCollapseMatterLambda.F90`

Subroutine: `sphericalCollapseMatterLambdaDereTabulate`

Description: Recompute the look-up tables for virial density contrast.

Code lines: 21

Contained by: file `structure_formation.virial_density_contrast.sphericalCollapseMatterLambda.F90`

Modules used: `sphericalCollapseMatterLambda`

Subroutine: `sphericalCollapseMatterLambdaDstateRestore`

Description: Reset the tabulation if state is to be retrieved. This will force tables to be rebuilt.

Code lines: 11

Contained by: file `structure_formation.virial_density_contrast.sphericalCollapseMatterLambda.F90`

Modules used: `fgsl`

Subroutine: `sphericalCollapseMatterLambdaDstateTestRestore`

Description: Reset the tabulation if state is to be retrieved. This will force tables to be rebuilt.

Code lines: 10
19.1. Program units

**Contained by:** file `structure_formation.virial_density_contrast.spherical_collapse_matter_lambda.F90`

**Modules used:** `fgsl`

**interface:** `virial_density_contrast.spherical_collapse_matter_lambda`

- **Description:** Constructors for the `sphericalCollapseMatterLambda` dark matter halo virial density contrast class.
- **Code lines:** 3
- **Contained by:** file `structure_formation.virial_density_contrast.spherical_collapse_matter_lambda.F90`

**file:** `system.command.F90`

- **Description:** Contains a module which executes system commands.
- **Code lines:** 46

**module:** `system_command`

- **Description:** Executes system commands.
- **Code lines:** 26
- **Contained by:** file `system.command.F90`
- **Used by:**
  - function `hopkins2007constructor`
  - subroutine `chemical_state_atomic_cie_cloudy_create`
  - subroutine `galacticus_output_close_file`
  - subroutine `geometry_survey_window_functions_caputi_2011_ukidss_uds`
  - subroutine `geometry_survey_window_functions_li_white_2009_sdss`
  - subroutine `geometry_survey_window_functions_martin_2010_alfalfa`
  - function `recfastconstructor`
  - subroutine `stellar_population_spectra_conroy_initialize_imf`
  - subroutine `transfer_function_camb_make`

**subroutine:** `system_command_do`

- **Description:** Executes the system command `command`, optionally returning the resulting status in `iStatus`.
- **Code lines:** 16
- **Contained by:** module `system_command`
- **Modules used:** `galacticus_error iso_varying_string`

**file:** `system.load.F90`

- **Description:** Contains a module which reports system load averages.
- **Code lines:** 67

**module:** `system_load`

- **Description:** Reports system load averages.
- **Code lines:** 47
- **Contained by:** file `system.load.F90`
- **Used by:** function `galacticus_task_evolve_tree`

**subroutine:** `system_load_get`

- **Description:** Reports on system load via `/proc/loadavg`.
- **Code lines:** 17
19. Source Code Documentation

**module** system_load

**function**: system_processor_count

*Description*: Return a count of the number of available processors.

*Code lines*: 18

*Contained by*: module system_load

**file**: tests.IO.HDF5.F90

*Code lines*: 682

**program**: tests_io_hdf5

*Description*: Tests the HDF5 I/O module.

*Code lines*: 664

*Contained by*: file tests.IO.HDF5.F90

**Modules used**: hdf5
iso_varying_string  kind_numbers
memory_management  unit_tests

**file**: tests.IO.XML.F90

*Description*: Contains a program which tests functionality of the XML I/O module.

*Code lines*: 87

**program**: tests_io_xml

*Description*: Tests the XML I/O module.

*Code lines*: 67

*Contained by*: file tests.IO.XML.F90

**Modules used**: foxDom
memory_management  unit_tests

**file**: tests.NFW96_concentration.dark_energy.F90

*Description*: Contains a program which tests the Navarro et al. [1996] halo concentration algorithm in a dark energy Universe. Comparisons are made to the “charden” code written by Julio Navarro.

*Code lines*: 99

**program**: test_nfw96_concentration_dark_energy

*Description*: Tests the Navarro et al. [1996] halo concentration algorithm in a dark energy Universe. Comparisons are made to the “charden” code written by Julio Navarro.

*Code lines*: 77

*Contained by*: file tests.NFW96_concentration.dark_energy.F90

**Modules used**: cosmology_functions  cosmology_parameters
dark_matter_profiles_concentration  galacticus_nodes
input_parameters  iso_varying_string
memory_management  string_handling
unit_tests

**file**: tests.ODE_solver.F90

*Description*: Contains a program to test ODE solver routines.

*Code lines*: 75
19.1. Program units

**program:** test_ode_solver

*Description:* Tests that ODE solver routines work.

*Code lines:* 55

*Contained by:* file tests.ODE_solver.F90

*Modules used:*
- fgs1
- iso_c_binding
- ode_solver
- test_ode_solver_functions
- unit_tests

**file:** tests.ODE_solver.functions.F90

*Description:* Contains a module of ODEs for unit tests.

*Code lines:* 55

**module:** test_ode_solver_functions

*Description:* Contains ODEs for unit tests.

*Code lines:* 35

*Contained by:* file tests.ODE_solver.functions.F90

*Modules used:*
- fgs1
- iso_c_binding

*Used by:* program test_ode_solver

**function:** ode_set_1

*Description:* A set of ODEs for unit tests.

*Code lines:* 10

*Contained by:* module test_ode_solver_functions

**function:** ode_set_2

*Description:* A set of ODEs for unit tests.

*Code lines:* 11

*Contained by:* module test_ode_solver_functions

**file:** tests.Prada2011_concentration.F90

*Description:* Contains a program which tests the Prada et al. [2011] halo concentration algorithm.

*Code lines:* 92

**program:** test_prada2011_concentration

*Description:* Tests the Prada et al. [2011] halo concentration algorithm. Values of concentration were read from their Figure 12.

*Code lines:* 72

*Contained by:* file tests.Prada2011_concentration.F90

*Modules used:*
- cosmology_functions
- dark_matter_profiles_concentration
- input_parameters
- memory_management
- cosmology_parameters
- galacticus_nodes
- iso_varying_string
- unit_tests

**file:** tests.Zhao2009_algorithms.EdS.F90

*Description:* Contains a program which tests the Zhao et al. [2009] halo mass formation history and halo concentration algorithms in an Einstein-de Sitter Universe.

*Code lines:* 132

**program:** test_zhao2009_flat
**Description:** Tests the Zhao et al. [2009] halo mass formation history and halo concentration algorithms in an Einstein-de Sitter Universe. Comparisons are made to the “mandc”. Note that comparison tolerances are relatively large since we have not attempted to match details (such as critical density calculation) with “mandc”.

**Code lines:** 111

**Contained by:** file tests.Zhao2009_algorithms.EdS.F90

**Modules used:**
- cosmology_functions
- dark_matter_profiles_concentration
- galacticus_input_paths
- input_parameters
- memory_management
- unit_tests
- dark_matter_halo_mass_accretion_histories
- file_utilities
- galacticus_nodes
- iso_varying_string
- string_handling

**file:** tests.Zhao2009_algorithms.dark_energy.F90

**Description:** Contains a program which tests the Zhao et al. [2009] halo mass formation history and halo concentration algorithms in a dark energy Universe.

**Code lines:** 133

**program:** test_zhao2009_dark_energy

**Description:** Tests the Zhao et al. [2009] halo mass formation history and halo concentration algorithms in a dark energy Universe. Comparisons are made to the “mandc”. Note that comparison tolerances are relatively large since we have not attempted to match details (such as critical density calculation) with “mandc”.

**Code lines:** 112

**Contained by:** file tests.Zhao2009_algorithms.dark_energy.F90

**Modules used:**
- cosmology_functions
- dark_matter_profiles_concentration
- galacticus_input_paths
- input_parameters
- memory_management
- unit_tests
- dark_matter_halo_mass_accretion_histories
- file_utilities
- galacticus_nodes
- iso_varying_string
- string_handling

**file:** tests.Zhao2009_algorithms.open.F90

**Description:** Contains a program which tests the Zhao et al. [2009] halo mass formation history and halo concentration algorithms in an open Universe.

**Code lines:** 133

**program:** test_zhao2009_open

**Description:** Tests the Zhao et al. [2009] halo mass formation history and halo concentration algorithms in an open Universe. Comparisons are made to the “mandc”. Note that comparison tolerances are relatively large since we have not attempted to match details (such as critical density calculation) with “mandc”.

**Code lines:** 112

**Contained by:** file tests.Zhao2009_algorithms.open.F90

**Modules used:**
- cosmology_functions
- dark_matter_profiles_concentration
- galacticus_input_paths
- input_parameters
- unit_tests
- dark_matter_halo_mass_accretion_histories
- file_utilities
- galacticus_nodes
- iso_varying_string
- string_handling
19.1. Program units

memory_management  string_handling
unit_tests

file: tests.abundances.F90
Description: Contains a program to test abundances objects functions.
Code lines: 56

program: test_abundances
Description: Test abundances objects.
Code lines: 36
Contained by: file tests.abundances.F90
Modules used: abundances_structure  input_parameters
iso_varying_string  memory_management
unit_tests

file: tests.arrays.F90
Description: Contains a program to test the array functions.
Code lines: 191

program: test_array_monotonicity
Description: Tests that array functions.
Code lines: 171
Contained by: file tests.arrays.F90
Modules used: array_utilities  iso_varying_string
kind_numbers  unit_tests

file: tests.black_hole_fundamentals.F90
Description: Contains a program to test the black hole fundamental functions.
Code lines: 45

program: test_black_hole_fundamentals
Description: Tests of black hole fundamental functions.
Code lines: 25
Contained by: file tests.black_hole_fundamentals.F90
Modules used: black_hole_fundamentals  unit_tests

file: tests.bug745815.F90
Code lines: 93

program: tests_bug745815
Description: Tests for regression of Bug #745815 (http://bugs.launchpad.net/galacticus/+bug/745815):
Skipping of a node during a tree walk.
Code lines: 75
Contained by: file tests.bug745815.F90
Modules used: galacticus_nodes  input_parameters
iso_varying_string  kind_numbers
memory_management  unit_tests

file: tests.comoving_distance.F90
19. Source Code Documentation

Code lines: 95

**program: tests_comoving_distance**

*Description:* Tests comoving distance calculations for various universes. Distances calculated using Python implementation of Ned Wright’s cosmology calculator.

*Code lines:* 77

*Contained by:* file `tests.comoving_distance.F90`

*Modules used:* cosmology_functions, cosmology_functions_options, cosmology_parameters, input_parameters, iso_varying_string, memory_management, unit_tests

**file: tests.comparisons.F90**

*Description:* Contains a program to test numerical comparison functions.

*Code lines:* 39

**program: test_comparison**

*Description:* Tests that numerical comparison functions work.

*Code lines:* 19

*Contained by:* file `tests.comparisons.F90`

*Modules used:* numerical_comparison, unit_tests

**file: tests.cosmic_age.F90**

*Code lines:* 108

**program: tests_cosmic_age**

*Description:* Tests cosmic age calculations for various Universes. Ages calculated using Python implementation of Ned Wright’s cosmology calculator.

*Code lines:* 90

*Contained by:* file `tests.cosmic_age.F90`

*Modules used:* cosmology_functions, cosmology_parameters, input_parameters, iso_varying_string, memory_management, numerical_constants_math, unit_tests

**file: tests.geometry.coordinate_systems.F90**

*Description:* Contains a program to coordinate system functions.

*Code lines:* 49

**program: test_coordinate_systems**

*Description:* Tests of coordinate system functions.

*Code lines:* 29

*Contained by:* file `tests.geometry.coordinate_systems.F90`

*Modules used:* coordinate_systems, numerical_constants_math, unit_tests

**file: tests.halo_mass_function.Tinker.F90**

*Description:* Contains a program which tests the Tinker et al. [2008] mass function by comparing to Jeremy Tinker’s code.

*Code lines:* 94
19.1. Program units

**program: tests_halo_mass_function_tinker**

*Description:* Tests the Tinker et al. [2008] mass function by comparing to Jeremy Tinker’s code.

*Code lines:* 73

*Contained by:* file `tests.halo_mass_function.Tinker.F90`

*Modules used:* cosmology_functions, critical_overdensity, halo_mass_function, iso_varying_string, memory_management, unit_tests

**file: tests.hashes.F90**

*Description:* Contains a program to test features of the hashes (i.e. associative arrays) module.

*Code lines:* 70

**program: test_hashes**

*Description:* Tests features of the hashes (i.e. associative arrays) module.

*Code lines:* 50

*Contained by:* file `tests.hashes.F90`

*Modules used:* hashes, memory_management, unit_tests

**file: tests.hashes.cryptographic.F90**

*Description:* Contains a program to test features of cryptographic hashes.

*Code lines:* 45

**program: test_hashes_cryptographic**

*Description:* Contains a program to test features of cryptographic hashes.

*Code lines:* 25

*Contained by:* file `tests.hashes.cryptographic.F90`

*Modules used:* hashes_cryptographic, iso_varying_string, memory_management, unit_tests

**file: tests.hashes.perfect.F90**

*Description:* Contains a program to test perfect hashing algorithms.

*Code lines:* 72

**program: test_perfect_hashes**

*Description:* Tests perfect hashing algorithms.

*Code lines:* 52

*Contained by:* file `tests.hashes.perfect.F90`

*Modules used:* hashes_perfect, kind_numbers, memory_management, unit_tests

**file: tests.integration.F90**

*Description:* Contains a program to test integration routines.

*Code lines:* 59

**program: test_integration**
19. Source Code Documentation

**Description:** Tests that numerical integration routines work.

**Code lines:** 39

**Contained by:** file `tests.integration.F90`

**Modules used:**
- iso_c_binding
- numerical_constants_math
- numerical_integration
- test_integration_functions
- unit_tests

**file:** `tests.integration.functions.F90`

**Description:** Contains a module of integrands for unit tests.

**Code lines:** 79

**module:** `test_integration_functions`

**Description:** Contains integrands for unit tests.

**Code lines:** 59

**Contained by:** file `tests.integration.functions.F90`

**Modules used:**
- f gsl
- iso_c_binding

**Used by:** program `test_integration`

**function:** `integrand1`

**Description:** Integral for unit testing.

**Code lines:** 9

**Contained by:** module `test_integration_functions`

**function:** `integrand2`

**Description:** Integral for unit testing.

**Code lines:** 9

**Contained by:** module `test_integration_functions`

**function:** `integrand3`

**Description:** Integral for unit testing.

**Code lines:** 9

**Contained by:** module `test_integration_functions`

**function:** `integrand4`

**Description:** Integral for unit testing.

**Code lines:** 10

**Contained by:** module `test_integration_functions`

**Modules used:**
- numerical_integration

**file:** `tests.interpolation.2D.F90`

**Description:** Contains a program to test 2D interpolation routines.

**Code lines:** 55

**program:** `test_interpolation_2d`

**Description:** Tests that 2D interpolation routines work.

**Code lines:** 35

**Contained by:** file `tests.interpolation.2D.F90`

**Modules used:**
- numerical_interpolation_2d_irregular
- unit_tests
19.1. Program units

file: tests.interpolation.F90
Description: Contains a program to test the numerical interpolation code.
Code lines: 60

program: test_interpolation
Description: Tests that numerical interpolation code works correctly.
Code lines: 40
Contained by: file tests.interpolation.F90
Modules used: fgs1

file: tests.kepler_orbits.F90
Code lines: 138

program: tests_kepler_orbits
Description: Tests for orbital parameter conversions.
Code lines: 120
Contained by: file tests.kepler_orbits.F90
Modules used: input_parameters iso_varying_string
kepler_orbits memory_management
numerical_constants_physical unit_tests

file: tests.linear_growth.EdS.F90
Code lines: 57

program: tests_linear_growth_eds
Description: Tests linear growth calculations.
Code lines: 39
Contained by: file tests.linear_growth.EdS.F90
Modules used: cosmology_functions input_parameters
iso_varying_string linear_growth
memory_management unit_tests

file: tests.linear_growth.cosmological_constant.F90
Code lines: 59

program: tests_linear_growth_cosmological_constant
Description: Tests linear growth calculations for a cosmological constant Universe. Growth rates are compared to calculations taken from Andrew Hamilton's "growl" code available at: http://casa.colorado.edu/ ajsh/growl/
Code lines: 41
Contained by: file tests.linear_growth.cosmological_constant.F90
Modules used: cosmology_functions input_parameters
iso_varying_string linear_growth
memory_management unit_tests

file: tests.linear_growth.dark_energy.F90
Code lines: 59
19. Source Code Documentation

program: tests_linear_growth_dark_energy
Description: Tests linear growth calculations for a dark energy Universe. Growth rates are compared to Figure 1 of Linder and Jenkins (2003; MNRAS; 346; 573; http://adsabs.harvard.edu/abs/2003MNRAS.346..573L).
Code lines: 41
Contained by: file tests.linear_growth.dark_energy.F90
Modules used: cosmology_functions input_parameters
iso_varying_string linear_growth
memory_management unit_tests

file: tests.linear_growth.open.F90
Description: Contains a program to test the numerical range making code.
Code lines: 55

program: tests_linear_growth_open
Description: Tests linear growth calculations for an open Universe. Growth rates are compared to calculations taken from: http://www.icosmos.co.uk/index.html
Code lines: 37
Contained by: file tests.linear_growth.open.F90
Modules used: cosmology_functions input_parameters
iso_varying_string linear_growth
unit_tests

file: tests.make_ranges.F90
Description: Contains a program to test mass distributions.
Code lines: 125

program: test_make_ranges
Description: Tests that numerical range making code works correctly.
Code lines: 32
Contained by: file tests.make_ranges.F90
Modules used: array_utilities numerical_ranges
unit_tests

file: tests.mass_distributions.F90
Description: Contains a program to test mass distributions.
Code lines: 105

program: test_mass_distributions
Description: Tests mass distributions.
Code lines: 105
Contained by: file tests.mass_distributions.F90
Modules used: coordinates mass_distributions
memory_management numerical_constants_math
unit_tests

file: tests.math_special_functions.F90
Description: Contains a program to test mathematical special functions.
Code lines: 92
19.1. Program units

program: test_math_special_functions
Description: Tests of mathematical special functions.
Code lines: 72
Contained by: file tests.math_special_functions.F90
Modules used: bessel_functions exponential_integrals
factorials gamma_functions
hypergeometric_functions unit_tests

file: tests.meshes.F90
Description: Contains a program to test the array functions.
Code lines: 70

program: test_meshes
Description: Test mesh functions.
Code lines: 50
Contained by: file tests.meshes.F90
Modules used: iso_c_binding meshes
unit_tests

file: tests.nodes.F90
Description: Contains a program which tests the nodes implementation.
Code lines: 163

program: test_nodes
Description: Tests the nodes implementation.
Code lines: 143
Contained by: file tests.nodes.F90
Modules used: array_utilities galacticus_error
galacticus_nodes input_parameters
iso_varying_string memory_management
test_nodes_tasks unit_tests

file: tests.nodes.task.F90
Description: Contains a module which implements a simple test of mapping a function over all components in a node.
Code lines: 82

module: test_nodes_tasks
Description: Implements a simple test of mapping a function over all components in a node.
Code lines: 62
Contained by: file tests.nodes.task.F90
Used by: program test_nodes

subroutine: test_node_task
Description: Implements simple tests of mapping functions over all components in a node.
Code lines: 25
Contained by: module test_nodes_tasks
Modules used: galacticus_nodes unit_tests
function: testfuncdouble0
  Description: A simple test function which returns the enclosed mass for a component. Used in testing mapping over a function over all components.
  Code lines: 10
  Contained by: module test_nodes_tasks
  Modules used: galactic_structure_options galacticus_nodes

subroutine: testvoidfunc
  Description: A simple void function used in testing mapping over a function over all components.
  Code lines: 10
  Contained by: module test_nodes_tasks
  Modules used: galacticus_nodes iso_varying_string

file: tests.power_spectrum.F90
  Description: Contains a program that tests power spectrum calculations.
  Code lines: 63

program: tests_power_spectrum
  Description: Tests power spectrum calculations.
  Code lines: 43
  Contained by: file tests.power_spectrum.F90
  Modules used: cosmology_parameters input_parameters
                   iso_varying_string memory_management
                   numerical_constants_math power_spectra
                   unit_tests

file: tests.random.F90
  Description: Contains a program to test random number functions.
  Code lines: 58

program: test_random
  Description: Tests that random number functions work.
  Code lines: 38
  Contained by: file tests.random.F90
  Modules used: fgsl
                   memory_management
                   pseudo_random
                   unit_tests

file: tests.regular_expressions.F90
  Description: Contains a program which tests regular expression functionality.
  Code lines: 40

program: tests_regular_expressions
  Description: Tests regular expression functionality.
  Code lines: 20
  Contained by: file tests.regular_expressions.F90
  Modules used: regular_expressions
                   unit_tests

file: tests.root_finding.F90

1906
19.1. Program units

**Description:** Contains a program to test root finding routines.

**Code lines:** 78

**program:** test_root_finding

**Description:** Tests that routine finding routines work.

**Code lines:** 58

**Contained by:** file tests.root_finding.F90

**Modules used:** root_finder  
unit_tests  
test_root_finding_functions

**file:** tests.root_finding_functions.F90

**Description:** Contains a module of functions for root finding unit tests.

**Code lines:** 49

**module:** test_root_finding_functions

**Description:** Contains functions for root finding unit tests.

**Code lines:** 29

**Contained by:** file tests.root_finding.functions.F90

**Used by:** program test_root_finding

**function:** root_function_1

**Description:** Function for root finding unit tests.

**Code lines:** 5

**Contained by:** module test_root_finding_functions

**function:** root_function_2

**Description:** Function for root finding unit tests.

**Code lines:** 5

**Contained by:** module test_root_finding_functions

**function:** root_function_3

**Description:** Function for root finding unit tests.

**Code lines:** 5

**Contained by:** module test_root_finding_functions

**file:** tests.search.F90

**Description:** Contains a program to test array search functions.

**Code lines:** 65

**program:** test_search

**Description:** Tests that array search functions work.

**Code lines:** 45

**Contained by:** file tests.search.F90

**Modules used:** arrays_search  
iso_varying_string  
kind_numbers  
unit_tests

**file:** tests.sigma.F90

**Code lines:** 73
program: tests_sigma
Description: Tests
Code lines: 55
Contained by: file tests_sigma.F90
Modules used: cosmology_parameters, iso_varying_string, numerical_constants_math, power_spectra, input_parameters, memory_management, numerical_ranges, unit_tests

file: tests.sort.F90
Description: Contains a program to test sorting functions.
Code lines: 52

program: test_sort
Description: Tests of sorting functions.
Code lines: 32
Contained by: file tests.sort.F90
Modules used: kind_numbers, sort, unit_tests

file: tests.spectra.postprocess.Inoue2014.F90
Description: Contains a program to test the Inoue et al. [2014] algorithm for IGM absorption.
Code lines: 53

program: test_inoue2014
Description: Tests the Inoue et al. [2014] algorithm for IGM absorption.
Code lines: 33
Contained by: file tests.spectra.postprocess.Inoue2014.F90
Modules used: numerical_constants_atomic, stellar_population_spectra_postprocess, unit_tests

file: tests.spherical_collapse.dark_energy.EdS.F90
Description: Contains a program which tests spherical collapse calculations for a dark energy Universe, specifically using an Einstein-de Sitter cosmology (e.g. Kitayama and Suto 1996; eqn. A2).
Code lines: 71

program: tests_spherical_collapse_dark_energy_eds
Description: Tests spherical collapse calculations for a dark energy Universe, specifically using an Einstein-de Sitter cosmology. Compares results to the analytic solution.
Code lines: 50
Contained by: file tests.spherical_collapse.dark_energy.EdS.F90
Modules used: cosmology_functions, input_parameters, memory_management, unit_tests, critical_overdensity, iso_varying_string, numerical_constants_math, virial_density_contrast

file: tests.spherical_collapse.dark_energy.constantEoSminus0.6.F90
19.1. Program units

Description: Contains a program which tests spherical collapse calculations for a dark energy Universe, specifically using a flat, $\omega = -0.6$ cosmology.

Code lines: 66

Program: tests_spherical_collapse_dark_energy_omega_zero_point_six
Description: Tests spherical collapse calculations for a dark energy Universe, specifically using a flat, $\omega = -0.6$ cosmology. Compares results to points read from Figure 6 of Horellou and Berge [2005] using DataThief.

Code lines: 45
Contained by: file tests.spherical_collapse.dark_energy.constantEoSminus0.6.F90
Modules used: cosmology_functions input_parameters
iso_varying_string memory_management
unit_tests virial_density_contrast

File: tests.spherical_collapse.dark_energy.constantEoSminus0.8.F90
Description: Contains a program which tests spherical collapse calculations for a dark energy Universe, specifically using a flat, $\omega = -0.8$ cosmology.

Code lines: 66

Program: tests_spherical_collapse_dark_energy_omega_zero_point_eight
Description: Tests spherical collapse calculations for a dark energy Universe, specifically using a flat, $\omega = -0.8$ cosmology. Compares results to points read from Figure 6 of Horellou and Berge [2005] using DataThief.

Code lines: 45
Contained by: file tests.spherical_collapse.dark_energy.constantEoSminus0.8.F90
Modules used: cosmology_functions input_parameters
iso_varying_string memory_management
unit_tests virial_density_contrast

File: tests.spherical_collapse.dark_energy.constantEoSminusHalf.F90
Description: Contains a program which tests spherical collapse calculations for a dark energy Universe, specifically using a flat, $\omega = -1/2$ cosmology.

Code lines: 67

Program: tests_spherical_collapse_dark_energy_omega_half
Description: Tests spherical collapse calculations for a dark energy Universe, specifically using a flat, $\omega = -1/2$ cosmology. Compares results to the fitting function of Weinberg and Kamionkowski (2003; eqn. 18).

Code lines: 46
Contained by: file tests.spherical_collapse.dark_energy.constantEoSminusHalf.F90
Modules used: cosmology_functions critical_overdensity
input_parameters iso_varying_string
linear_growth memory_management
numerical_constants_math unit_tests

File: tests.spherical_collapse.dark_energy.constantEoSminusTwoThirds.F90
Description: Contains a program which tests spherical collapse calculations for a dark energy Universe, specifically using a flat, $\omega = -2/3$ cosmology.

Code lines: 68
program: tests_spherical_collapse_dark_energy_omega_two_thirds
Description: Tests spherical collapse calculations for a dark energy Universe, specifically using a flat, $\omega = -2/3$ cosmology. Compares results to the fitting function of Weinberg and Kamionkowski (2003; eqn. 18).
Code lines: 47
Contained by: file tests.spherical_collapse.dark_energy.constantEoSminusTwoThirds.F90
Modules used: cosmology_functions critical_overdensity
input_parameters iso_varying_string
linear_growth memory_management
numerical_constants_math unit_tests

file: tests.spherical_collapse.dark_energy.lambda.F90
Description: Contains a program which tests spherical collapse calculations for a dark energy Universe, specifically using a flat, cosmological constant cosmology.
Code lines: 74

program: tests_spherical_collapse_dark_energy_lambda
Description: Tests spherical collapse calculations for a dark energy Universe, specifically using a flat, cosmological constant cosmology. Compares results to the fitting function of Kitayama and Suto (1996; eqn. A6).
Code lines: 53
Contained by: file tests.spherical_collapse.dark_energy.lambda.F90
Modules used: cosmology_functions critical_overdensity
input_parameters iso_varying_string
linear_growth memory_management
numerical_constants_math unit_tests
virial_density_contrast

file: tests.spherical_collapse.dark_energy.open.F90
Description: Contains a program which tests spherical collapse calculations for a dark energy Universe, specifically using an open cosmology.
Code lines: 66

program: tests_spherical_collapse_dark_energy_open
Description: Tests spherical collapse calculations for a dark energy Universe, specifically using an open cosmology. Compares results to the analytic solution (e.g. Kitayama and Suto 1996; eqn. A4).
Code lines: 45
Contained by: file tests.spherical_collapse.dark_energy.open.F90
Modules used: cosmology_functions critical_overdensity
input_parameters iso_varying_string
linear_growth memory_management
numerical_constants_math unit_tests

file: tests.spherical_collapse.flat.F90
Description: Contains a program which tests spherical collapse calculations for a flat Universe.
Code lines: 66
program: tests_sphericalCollapseFlat
Description: Tests spherical collapse calculations for a flat Universe. Compares results to the fitting formula of Bryan and Norman [1998].
Code lines: 46
Contained by: file tests.sphericalCollapse.flat.F90
Modules used: cosmology_functions
iso_varying_string
numerical_constants_math
virial_density_contrast

file: tests.sphericalCollapse.open.F90
Description: Contains a program which tests spherical collapse calculations for an open Universe.
Code lines: 66

program: tests_sphericalCollapseOpen
Description: Tests spherical collapse calculations for an open Universe. Compares results to the fitting formula of Bryan and Norman [1998].
Code lines: 46
Contained by: file tests.sphericalCollapse.open.F90
Modules used: cosmology_functions
iso_varying_string
numerical_constants_math
virial_density_contrast

file: tests.string_utilities.F90
Description: Contains a program to test string handling utilities
Code lines: 84

program: test_string_utilities
Description: Tests that numerical range making code works correctly.
Code lines: 64
Contained by: file tests.string_utilities.F90
Modules used: iso_varying_string
kind_numbers
string_handling

file: tests.tables.F90
Description: Contains a program to test tables.
Code lines: 109

program: test_tables
Description: Tests that tables work correctly.
Code lines: 89
Contained by: file tests.tables.F90
Modules used: array_utilities
memory_management
tables

file: tests.tensors.F90
Description: Contains a program to test tensor functionality.


19. Source Code Documentation

**Code lines:** 58

**program:** test_tensors

*Description:* Tests of coordinate system functions.

*Code lines:* 38

*Contained by:* file `tests.tensors.F90`

*Modules used:* tensors unit_tests

**file:** tests.tree_branch_destroy.F90

*Code lines:* 79

**program:** tests_tree_branch_destroy

*Code lines:* 61

*Contained by:* file `tests.tree_branch_destroy.F90`

*Modules used:* galacticus_nodes kind_numbers memory_management unit_tests

**file:** tests.vectors.F90

*Description:* Contains a program to test vector functions.

*Code lines:* 40

**program:** test_vectors

*Description:* Tests of vector functions.

*Code lines:* 20

*Contained by:* file `tests.vectors.F90`

*Modules used:* unit_tests vectors

**file:** thermodynamics.ideal_gases.F90

*Description:* Contains a module which implements thermodynamic properties of ideal gases.

*Code lines:* 59

**module:** ideal_gases_thermodynamics

*Description:* Implements thermodynamic properties of ideal gases.

*Code lines:* 39

*Contained by:* file `thermodynamics.ideal_gases.F90`

*Used by:* function bondi_hoyle_lyttleton_accretion_radius

*Modules used:* numerical_constants_physical

**function:** ideal_gas_jean_length

*Description:* Return the Jeans length (in Mpc) for gas of given temperature and density.

*Code lines:* 8

*Contained by:* module `ideal_gases_thermodynamics`

*Modules used:* numerical_constants_physical

**function:** ideal_gas_sound_speed

*Description:* Return the sound speed (in km/s) for an ideal gas of given temperature and (optionally) meanAtomicMass.

*Code lines:* 19
19.1. Program units

Contained by:  module ideal_gases_thermodynamics
Modules used: numerical_constants_astronomical  numerical_constants_physical

file: thermodynamics.radiation.F90
Code lines:  72

module: thermodynamics_radiation
Description: Implements calculations of thermal radiation.
Code lines:  54
Contained by:  file thermodynamics.radiation.F90
Used by:  subroutine radiation_flux cmb

function: blackbody_emission
Description: Compute the Planck blackbody spectral radiance (defined per unit wavelength, in units of 
  J s\(^{-1}\) m\(^{-2}\) sr\(^{-1}\) Å\(^{-1}\)) or J s\(^{-1}\) m\(^{-2}\) sr\(^{-1}\) Hz\(^{-1}\) depending on the optional radianceType 
  argument). Input wavelength is in Angstroms, input temperature is in Kelvin.
Code lines:  40
Contained by:  module thermodynamics_radiation
Modules used: numerical_constants_physical  numerical_constants_units

file: tidal_stripping.mass_loss_rate.disks.F90
Description: Contains a module that implements calculations of mass loss rates from disks due to tidal 
  stripping.
Code lines:  99

module: tidal_stripping_mass_loss_rate_disks
Description: Implements calculations of mass loss rates from disks due to tidal stripping.
Code lines:  78
Contained by:  file tidal_stripping.mass_loss_rate.disks.F90
Used by:  subroutine node_component_disk_-expontial_rate_compute

function: tidal_stripping_mass_loss_rate_disk
Description: Return the tidal force for the hot halo of thisNode.
Code lines:  62
Contained by:  module tidal_stripping_mass_loss_rate_disks
Modules used: galacticus_error  galacticus_nodes
  input_parameters  iso_varying_string
  tidal_stripping_mass_loss_rate_-disks_null  tidal_stripping_mass_loss_rate_-disks_simple

file: tidal_stripping.mass_loss_rate.disks.null.F90
Description: Contains a module which implements a null mass loss rates from disks due to tidal stripping.
Code lines:  54

module: tidal_stripping_mass_loss_rate_disks_null
Description: Implements a null mass loss rates from disks due to tidal stripping.
Code lines:  34
Contained by:  file tidal_stripping.mass_loss_rate.disks.null.F90
19. Source Code Documentation

**Modules used:** galacticus_nodes

**Used by:** function tidal_stripping_mass_loss_rate_disk

**function: tidal_stripping_mass_loss_rate_disk_null**

**Description:** Computes the mass loss rate from disks due to tidal stripping. Always returns zero.

**Code lines:** 8

**Contained by:** module tidal_stripping_mass_loss_rate_disks_null

**Modules used:** galacticus_nodes

**subroutine: tidal_stripping_mass_loss_rate_disks_null_init**

**Description:** Initializes the “null” tidal stripping mass loss rate from disks module.

**Code lines:** 10

**Contained by:** module tidal_stripping_mass_loss_rate_disks_null

**Modules used:** input_parameters iso_varying_string

**file: tidal_stripping_mass_loss_rate_disks.simple.F90**

**Description:** Contains a module which implements simple mass loss rates from disks due to tidal stripping.

**Code lines:** 120

**module: tidal_stripping_mass_loss_rate_disks_simple**

**Description:** Implements simple mass loss rates from disks due to tidal stripping.

**Code lines:** 100

**Contained by:** file tidal_stripping_mass_loss_rate_disks.simple.F90

**Modules used:** galacticus_nodes

**Used by:** function tidal_stripping_mass_loss_rate_disk_simple

**function: tidal_stripping_mass_loss_rate_disk_simple**

**Description:** Computes the mass loss rate from disks due to tidal stripping assuming a simple model. Specifically, the mass loss rate is

\[
\dot{M} = -\alpha M/\tau_{\text{disk}},
\]

(19.40)

where

\[
\alpha = F_{\text{tidal}}/F_{\text{gravity}},
\]

(19.41)

\[F_{\text{tidal}} = F_{\text{tidal}r_{1/2}}, \quad F_{\text{tidal}}\] is the tidal field from the host halo (see §13.35), and

\[
F_{\text{gravity}} = V_{1/2}^2(r_{1/2})/r_{1/2}
\]

(19.42)

is the gravitational restoring force in the disk at the half-mass radius, \(r_{1/2}\).

**Code lines:** 58

**Contained by:** module tidal_stripping_mass_loss_rate_disks_simple

**Modules used:** galacticus_structure_options galacticus_nodes galacticus_structure_rotation_curves numerical_constants_astronomical numerical_constants_math numerical_constants_physical satellites_tidal_fields

**subroutine: tidal_stripping_mass_loss_rate_disks_simple_init**
19.1. Program units

Description: Initializes the “simple” tidal stripping mass loss rate from disks module.
Code lines: 23
Contained by: module tidal_stripping_mass_loss_rate_disks_simple
Modules used: input_parameters iso_varying_string

file: tidal_stripping_mass_loss_rate.spheroids.F90
Description: Contains a module that implements calculations of mass loss rates from spheroids due to tidal stripping.
Code lines: 99

module: tidal_stripping_mass_loss_rate_spheroids
Description: Implements calculations of mass loss rates from spheroids due to tidal stripping.
Code lines: 78
Contained by: file tidal_stripping_mass_loss_rate.spheroids.F90
Used by: subroutine node_component_spheroid_standard_rate_compute

function: tidal_stripping_mass_loss_rate_spheroid
Description: Return the tidal force for the hot halo of thisNode.
Code lines: 62
Contained by: module tidal_stripping_mass_loss_rate_spheroids
Modules used: galacticus_error galacticus_nodes
input_parameters iso_varying_string tidal_stripping_mass_loss_rate_spheroids_null tidal_stripping_mass_loss_rate_spheroids_simple

file: tidal_stripping_mass_loss_rate.spheroids.null.F90
Description: Contains a module which implements a null mass loss rates from spheroids due to tidal stripping.
Code lines: 54

module: tidal_stripping_mass_loss_rate_spheroids_null
Description: Implements a null mass loss rates from spheroids due to tidal stripping.
Code lines: 34
Contained by: file tidal_stripping_mass_loss_rate.spheroids.null.F90
Modules used: galacticus_nodes

function: tidal_stripping_mass_loss_rate_spheroid_null
Description: Computes the mass loss rate from spheroids due to tidal stripping. Always returns zero.
Code lines: 8
Contained by: module tidal_stripping_mass_loss_rate_spheroids_null
Modules used: galacticus_nodes

subroutine: tidal_stripping_mass_loss_rate_spheroids_null_init
Description: Initializes the “null” tidal stripping mass loss rate from spheroids module.
Code lines: 10
Contained by: module tidal_stripping_mass_loss_rate_spheroids_null
19. Source Code Documentation

**Modules used:** input_parameters, iso_varying_string

**file:** tidal_stripping.mass_loss_rate.spheroids.simple.F90

**Description:** Contains a module which implements simple mass loss rates from spheroids due to tidal stripping.

**Code lines:** 120

**module:** tidal_stripping_mass_loss_rate_spheroids_simple

**Description:** Implements simple mass loss rates from spheroids due to tidal stripping.

**Code lines:** 100

**Contained by:** file tidal_stripping.mass_loss_rate.spheroids.simple.F90

**Modules used:** galacticus_nodes

**Used by:** function tidal_stripping.mass_loss_rate.spheroids.simple

**function:** tidal_stripping_mass_loss_rate_spheroids_simple

**Description:** Computes the mass loss rate from spheroids due to tidal stripping assuming a simple model. Specifically, the mass loss rate is

\[ \dot{M} = -\alpha M / \tau_{\text{spheroid}}, \]

where

\[ \alpha = F_{\text{tidal}} / F_{\text{gravity}}, \]

\[ F_{\text{tidal}} = \mathcal{T}_{\text{tidal}} r_{1/2}, \]

\[ F_{\text{gravity}} = V_{1/2}^2 (r_{1/2}) / r_{1/2} \]

is the gravitational restoring force in the spheroid at the half-mass radius, \( r_{1/2} \).

**Code lines:** 58

**Contained by:** module tidal_stripping_mass_loss_rate_spheroids_simple

**Modules used:** galacticus_nodes, numerical_constants_astronomical, numerical_constants_math, numerical_constants_physical, satellites_tidal_fields

**subroutine:** tidal_stripping_mass_loss_rate_spheroids_simple_init

**Description:** Initializes the “simple” tidal stripping mass loss rate from spheroids module.

**Code lines:** 23

**Contained by:** module tidal_stripping_mass_loss_rate_spheroids_simple

**Modules used:** input_parameters, iso_varying_string

**file:** utility.IO.HDF5.F90

**Description:** Contains a module that implements simple and convenient interfaces to a variety of HDF5 functionality.

**Code lines:** 10619

**module:** io_hdf5

**Description:** Implements simple and convenient interfaces to a variety of HDF5 functionality.

**Code lines:** 10596
19.1. Program units

<table>
<thead>
<tr>
<th>Contained by:</th>
<th>Modules used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>file utility.IO.HDF5.F90</td>
<td>hdf5</td>
</tr>
<tr>
<td>iso_varying_string</td>
<td>iso_c_binding</td>
</tr>
</tbody>
</table>

| Used by: | | |
|-----------|---------|
| program conditional_mass_function | program tests_excursion_sets |
| program mass_function_covariance | program xray_absorption.ism.wilms2000 |
| subroutine fileloadfile | subroutine fileloadfile |
| module galacticus_hdf5 | subroutine galacticus_output_analysis_mass_functions |
| subroutine galacticus_growth_factor_output | subroutine galacticus_version_output |
| subroutine geometry_survey_window_functions_caputi_2011.ukidss.uds | module halo_mass_function_tasks |
| function intergalactic_medium_state_internal_update | subroutine merger_tree_conditional_mass_function_output |
| subroutine merger_tree_build_initialize | module merger_tree_read_importers |
| file merger_trees.construct.read.importer.SussingMergers | file merger_trees.construct.read.importer.galacticus.F90 |
| module merger_tree_mass_accretion_history | module merger_tree_output_structure |
| subroutine merger_trees_render_dump | subroutine merger_tree_data_structure_export_galacticus |
| subroutine merger_tree_data_structure_export_irate | subroutine store_unit_attributes_galacticus |
| subroutine store_unit_attributes_irate | module galacticus_nodes |
| subroutine node_component_dark_matter_profile_scale_tree_output | subroutine node_component_dark_matter_profile_scale_shape_tree_output |
| subroutine node_component_dynamics_statistics_bars_output | program optimal_sampling_smf |
| module power_spectrum_tasks | function radiation_intergalactic_background_internal_update |
| subroutine stellar_tracks_initialize_file | function stellar_population_luminosity |
| subroutine stellar_population_spectra_conroy_initialize_imf | |
| subroutine excursion_sets_first_crossing_farahi_read_file | subroutine excursion_sets_first_crossing_farahi_write_file |
| program tests_io_hdf5 | module input_parameters |

**type: hdf5object**

**Description:** A structure that holds properties of HDF5 objects.

**Code lines:** 258

**Contained by:** module io_hdf5

**subroutine: io_hdf5_assert_attribute_type**

**Description:** Asserts that an attribute is of a certain type and rank.

**Code lines:** 75

**Contained by:** module io_hdf5

**Modules used:** galacticus_error
19. Source Code Documentation

**subroutine: io_hdf5_assert_dataset_type**
*Description:* Asserts that an dataset is of a certain type and rank.
*Code lines:* 61
*Contained by:* module `io_hdf5`
*Modules used:* `galacticus_error`

**function: io_hdf5_character_types**
*Description:* Return datatypes for character data of a given length. Types are for Fortran native and C native types.
*Code lines:* 30
*Contained by:* module `io_hdf5`
*Modules used:* `galacticus_error`

**subroutine: io_hdf5_close**
*Description:* Close an HDF5 object.
*Code lines:* 89
*Contained by:* module `io_hdf5`
*Modules used:* `galacticus_display`, `galacticus_error`, `string_handling`

**subroutine: io_hdf5_create_reference_scalar_to_1d**
*Description:* Create a scalar reference to the 1-D `toDataset` in the HDF5 group `fromGroup`.
*Code lines:* 111
*Contained by:* module `io_hdf5`
*Modules used:* `galacticus_error`

**subroutine: io_hdf5_create_reference_scalar_to_2d**
*Description:* Create a scalar reference to the 2-D `toDataset` in the HDF5 group `fromGroup`.
*Code lines:* 111
*Contained by:* module `io_hdf5`
*Modules used:* `galacticus_error`

**subroutine: io_hdf5_create_reference_scalar_to_3d**
*Description:* Create a scalar reference to the 3-D `toDataset` in the HDF5 group `fromGroup`.
*Code lines:* 111
*Contained by:* module `io_hdf5`
*Modules used:* `galacticus_error`

**subroutine: io_hdf5_create_reference_scalar_to_4d**
*Description:* Create a scalar reference to the 4-D `toDataset` in the HDF5 group `fromGroup`.
*Code lines:* 111
*Contained by:* module `io_hdf5`
*Modules used:* `galacticus_error`

**subroutine: io_hdf5_create_reference_scalar_to_5d**
*Description:* Create a scalar reference to the 5-D `toDataset` in the HDF5 group `fromGroup`.
*Code lines:* 111
*Contained by:* module `io_hdf5`
19.1. Program units

 Modules used: galacticus_error

 function: io_hdf5_dataset_rank
 Description: Return the rank of dataset datasetObject.
 Code lines: 39
 Contained by: module io_hdf5
 Modules used: galacticus_error

 function: io_hdf5_dataset_size
 Description: Return the size of the $\text{dim}^{th}$ dimension of dataset datasetObject.
 Code lines: 63
 Contained by: module io_hdf5
 Modules used: galacticus_error

 subroutine: io_hdf5_destroy
 Description: Destroy an HDF5 object by destroying its associated varying string objects.
 Code lines: 8
 Contained by: module io_hdf5

 function: io_hdf5_has_attribute
 Description: Check if thisObject has an attribute with the given attributeName.
 Code lines: 25
 Contained by: module io_hdf5
 Modules used: galacticus_error

 function: io_hdf5_has_dataset
 Description: Check if thisObject has a dataset with the given datasetName.
 Code lines: 40
 Contained by: module io_hdf5
 Modules used: galacticus_error

 function: io_hdf5_has_group
 Description: Check if thisObject has a group with the given groupName.
 Code lines: 32
 Contained by: module io_hdf5
 Modules used: galacticus_error

 subroutine: io_hdf5_initialize
 Description: Initialize the HDF5 subsystem.
 Code lines: 22
 Contained by: module io_hdf5
 Modules used: galacticus_error

 function: io_hdf5_is_open
 Description: Returns true if thisObject is open.
 Code lines: 7
 Contained by: module io_hdf5

 function: io_hdf5_is_reference
19. Source Code Documentation

Description: Return true if the input dataset is a scalar reference.
Code lines: 37
Contained by: module io_hdf5
Modules used: galacticus_error

**function: io_hdf5_open_attribute**

Description: Open an attribute in inObject.
Code lines: 116
Contained by: module io_hdf5
Modules used: galacticus_error

**function: io_hdf5_open_dataset**

Description: Open an dataset in inObject.
Code lines: 248
Contained by: module io_hdf5
Modules used: galacticus_error

**subroutine: io_hdf5_open_file**

Description: Open a file and return an appropriate HDF5 object. The file name can be provided as an input parameter or, if not provided, will be taken from the stored object name in fileObject.
Code lines: 144
Contained by: module io_hdf5
Modules used: file_utilities galacticus_error

**function: io_hdf5_open_group**

Description: Open an HDF5 group and return an appropriate HDF5 object. The group name can be provided as an input parameter or, if not provided, will be taken from the stored object name in groupObject. The location at which to open the group is taken from either inObject or inPath.
Code lines: 112
Contained by: module io_hdf5
Modules used: galacticus_error

**function: io_hdf5_path_to**

Description: Returns the path to thisObject.
Code lines: 8
Contained by: module io_hdf5

**subroutine: io_hdf5_read_attribute_character_1d_array_allocatable**

Description: Open and read an character scalar attribute in thisObject.
Code lines: 107
Contained by: module io_hdf5
Modules used: galacticus_error memory_management

**subroutine: io_hdf5_read_attribute_character_1d_array_static**

Description: Open and read an character scalar attribute in thisObject.
Code lines: 108
Contained by: module io_hdf5
Modules used: galacticus_error
**subroutine: io_hdf5_read_attribute_character_scalar**
- **Description:** Open and read an character scalar attribute in thisObject.
- **Code lines:** 127
- **Contained by:** module io_hdf5
- **Modules used:** galacticus_error

**subroutine: io_hdf5_read_attribute_double_1d_array_allocatable**
- **Description:** Open and read an double scalar attribute in thisObject.
- **Code lines:** 92
- **Contained by:** module io_hdf5
- **Modules used:** galacticus_error memory_management

**subroutine: io_hdf5_read_attribute_double_1d_array_static**
- **Description:** Open and read an double scalar attribute in thisObject.
- **Code lines:** 93
- **Contained by:** module io_hdf5
- **Modules used:** galacticus_error

**subroutine: io_hdf5_read_attribute_double_scalar**
- **Description:** Open and read an double scalar attribute in thisObject.
- **Code lines:** 112
- **Contained by:** module io_hdf5
- **Modules used:** galacticus_error

**subroutine: io_hdf5_read_attribute_integer8_1d_array_allocatable**
- **Description:** Open and read an integer scalar attribute in thisObject.
- **Code lines:** 95
- **Contained by:** module io_hdf5
- **Modules used:** galacticus_error kind_numbers memory_management

**subroutine: io_hdf5_read_attribute_integer8_1d_array_static**
- **Description:** Open and read an integer scalar attribute in thisObject.
- **Code lines:** 102
- **Contained by:** module io_hdf5
- **Modules used:** galacticus_error kind_numbers memory_management

**subroutine: io_hdf5_read_attribute_integer8_scalar**
- **Description:** Open and read a long integer scalar attribute in thisObject.
- **Code lines:** 113
- **Contained by:** module io_hdf5
- **Modules used:** galacticus_error kind_numbers

**subroutine: io_hdf5_read_attribute_integer_1d_array_allocatable**
- **Description:** Open and read an integer scalar attribute in thisObject.
- **Code lines:** 92
- **Contained by:** module io_hdf5
subroutine: io_hdf5_read_attribute_integer_1d_array_static
Description: Open and read an integer scalar attribute in thisObject.
Code lines: 93
Contained by: module io_hdf5
Modules used: galacticus_error

subroutine: io_hdf5_read_attribute_integer_scalar
Description: Open and read an integer scalar attribute in thisObject.
Code lines: 110
Contained by: module io_hdf5
Modules used: galacticus_error

subroutine: io_hdf5_read_attribute_varstring_1d_array_allocatable
Description: Open and read a varying string 1-D array attribute in thisObject.
Code lines: 84
Contained by: module io_hdf5
Modules used: galacticus_error

subroutine: io_hdf5_read_attribute_varstring_1d_array_allocatable_do_read
Description: Open and read a varying string 1-D array attribute in thisObject by creating a suitably-sized character variable into which it can be read.
Code lines: 21
Contained by: module io_hdf5
Modules used: memory_management

subroutine: io_hdf5_read_attribute_varstring_1d_array_static
Description: Open and read a varying string 1-D array attribute in thisObject.
Code lines: 84
Contained by: module io_hdf5
Modules used: galacticus_error

subroutine: io_hdf5_read_attribute_varstring_1d_array_static_do_read
Description: Open and read a varying string 1-D array attribute in thisObject by creating a suitably-sized character variable into which it can be read.
Code lines: 17
Contained by: module io_hdf5

subroutine: io_hdf5_read_attribute_varstring_scalar
Description: Open and read a varying string scalar attribute in thisObject.
Code lines: 85
Contained by: module io_hdf5
Modules used: galacticus_error

subroutine: io_hdf5_read_attribute_varstring_scalar_do_read
Description: Open and read a varying string scalar attribute in thisObject by creating a suitably-sized character variable into which it can be read.
Code lines: 18
Contained by: module io_hdf5

1922
subsection: io_hdf5_read_dataset_character_1d_array_allocatable
Description: Open and read an integer scalar dataset in thisObject.
Code lines: 294
Contained by: module io_hdf5
Modules used: galacticus_error memory_management

subsection: io_hdf5_read_dataset_character_1d_array_static
Description: Open and read a character scalar dataset in thisObject.
Code lines: 295
Contained by: module io_hdf5
Modules used: galacticus_error

subsection: io_hdf5_read_dataset_double_1d_array_allocatable
Description: Open and read a double scalar dataset in thisObject.
Code lines: 286
Contained by: module io_hdf5
Modules used: galacticus_error memory_management

subsection: io_hdf5_read_dataset_double_1d_array_static
Description: Open and read a double scalar dataset in thisObject.
Code lines: 280
Contained by: module io_hdf5
Modules used: galacticus_error

subsection: io_hdf5_read_dataset_double_2d_array_allocatable
Description: Open and read a double 2-D array dataset in thisObject.
Code lines: 280
Contained by: module io_hdf5
Modules used: galacticus_error memory_management

subsection: io_hdf5_read_dataset_double_2d_array_static
Description: Open and read a double 2-D array dataset in thisObject.
Code lines: 281
Contained by: module io_hdf5
Modules used: galacticus_error

subsection: io_hdf5_read_dataset_double_3d_array_allocatable
Description: Open and read a double 3-D array dataset in thisObject.
Code lines: 280
Contained by: module io_hdf5
Modules used: galacticus_error memory_management

subsection: io_hdf5_read_dataset_double_3d_array_static
Description: Open and read a double 3-D array dataset in thisObject.
Code lines: 281
Contained by: module io_hdf5
Modules used: galacticus_error
subroutine: io_hdf5_read_dataset_double_4d_array_allocatable
Description: Open and read a double 4-D array dataset in thisObject.
Code lines: 280
Contained by: module io_hdf5
Modules used: galacticus_error memory_management

subroutine: io_hdf5_read_dataset_double_4d_array_static
Description: Open and read a double scalar dataset in thisObject.
Code lines: 281
Contained by: module io_hdf5
Modules used: galacticus_error

subroutine: io_hdf5_read_dataset_double_5d_array_allocatable
Description: Open and read a double 5-D array dataset in thisObject.
Code lines: 279
Contained by: module io_hdf5
Modules used: galacticus_error memory_management

subroutine: io_hdf5_read_dataset_double_5d_array_static
Description: Open and read a double scalar dataset in thisObject.
Code lines: 281
Contained by: module io_hdf5
Modules used: galacticus_error

subroutine: io_hdf5_read_dataset_integer8_1d_array_allocatable
Description: Open and read a long integer scalar dataset in thisObject.
Code lines: 282
Contained by: module io_hdf5
Modules used: galacticus_error kind_numbers memory_management

subroutine: io_hdf5_read_dataset_integer8_1d_array_static
Description: Open and read a long integer scalar dataset in thisObject.
Code lines: 288
Contained by: module io_hdf5
Modules used: galacticus_error kind_numbers memory_management

subroutine: io_hdf5_read_dataset_integer_1d_array_allocatable
Description: Open and read an integer scalar dataset in thisObject.
Code lines: 280
Contained by: module io_hdf5
Modules used: galacticus_error memory_management

subroutine: io_hdf5_read_dataset_integer_1d_array_static
Description: Open and read an integer scalar dataset in thisObject.
Code lines: 281
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Code lines</th>
<th>Contained by</th>
<th>Modules used</th>
</tr>
</thead>
<tbody>
<tr>
<td>subroutine: io_hdf5_read_dataset_varstring_1d_array_allocatable</td>
<td>Open and read an varying string 1-D array dataset in thisObject.</td>
<td>84</td>
<td>module io_hdf5</td>
<td>galacticus_error</td>
</tr>
<tr>
<td>subroutine: io_hdf5_read_dataset_varstring_1d_array_allocatable_do_read</td>
<td>Open and read an varying string 1-D array dataset in thisObject by creating a suitably-sized character variable into which it can be read.</td>
<td>21</td>
<td>module io_hdf5</td>
<td>memory_management</td>
</tr>
<tr>
<td>subroutine: io_hdf5_read_dataset_varstring_1d_array_static</td>
<td>Open and read an varying string 1-D array dataset in thisObject.</td>
<td>84</td>
<td>module io_hdf5</td>
<td>galacticus_error</td>
</tr>
<tr>
<td>subroutine: io_hdf5_read_dataset_varstring_1d_array_static_do_read</td>
<td>Open and read an varying string 1-D array dataset in thisObject by creating a suitably-sized character variable into which it can be read.</td>
<td>17</td>
<td>module io_hdf5</td>
<td>galacticus_error</td>
</tr>
<tr>
<td>subroutine: io_hdf5_set_defaults</td>
<td>Sets the compression level and chunk size for dataset output.</td>
<td>19</td>
<td>module io_hdf5</td>
<td>galacticus_error</td>
</tr>
<tr>
<td>subroutine: io_hdf5_uninitialize</td>
<td>Uninitialize the HDF5 subsystem.</td>
<td>15</td>
<td>module io_hdf5</td>
<td>galacticus_error</td>
</tr>
<tr>
<td>subroutine: io_hdf5_write_attribute_character_1d</td>
<td>Open and write an character 1-D array attribute in thisObject.</td>
<td>90</td>
<td>module io_hdf5</td>
<td>galacticus_error</td>
</tr>
<tr>
<td>subroutine: io_hdf5_write_attribute_character_scalar</td>
<td>Open and write an character scalar attribute in thisObject.</td>
<td>96</td>
<td>module io_hdf5</td>
<td>galacticus_error</td>
</tr>
</tbody>
</table>
19. Source Code Documentation

Modules used: galacticus_error

subroutine: io_hdf5_write_attribute_double_1d
Description: Open and write an double 1-D array attribute in thisObject.
Code lines: 77
Contained by: module io_hdf5
Modules used: galacticus_error

subroutine: io_hdf5_write_attribute_double_scalar
Description: Open and write an double scalar attribute in thisObject.
Code lines: 76
Contained by: module io_hdf5
Modules used: galacticus_error

subroutine: io_hdf5_write_attribute_integer8_1d
Description: Open and write an integer 1-D array attribute in thisObject.
Code lines: 87
Contained by: module io_hdf5
Modules used: galacticus_error, kind_numbers, memory_management

subroutine: io_hdf5_write_attribute_integer8_scalar
Description: Open and write a long integer scalar attribute in thisObject.
Code lines: 78
Contained by: module io_hdf5
Modules used: galacticus_error, kind_numbers

subroutine: io_hdf5_write_attribute_integer_1d
Description: Open and write an integer 1-D array attribute in thisObject.
Code lines: 77
Contained by: module io_hdf5
Modules used: galacticus_error

subroutine: io_hdf5_write_attribute_integer_scalar
Description: Open and write an integer scalar attribute in thisObject.
Code lines: 74
Contained by: module io_hdf5
Modules used: galacticus_error

subroutine: io_hdf5_write_attribute_varstring_1d
Description: Open and write a varying string 1-D array attribute in thisObject.
Code lines: 12
Contained by: module io_hdf5
Modules used: string_handling

subroutine: io_hdf5_write_attribute_varstring_scalar
Description: Open and write a varying string scalar attribute in thisObject.
Code lines: 11
Contained by: module io_hdf5
**subroutine: io_hdf5_write_dataset_character_1d**
*Description:* Open and write a character 1-D array dataset in thisObject.
*Code lines:* 178
*Contained by:* module io_hdf5
*Modules used:* galacticus_error

**subroutine: io_hdf5_write_dataset_double_1d**
*Description:* Open and write a double 1-D array dataset in thisObject.
*Code lines:* 159
*Contained by:* module io_hdf5
*Modules used:* galacticus_error

**subroutine: io_hdf5_write_dataset_double_2d**
*Description:* Open and write a double 2-D array dataset in thisObject.
*Code lines:* 171
*Contained by:* module io_hdf5
*Modules used:* galacticus_error

**subroutine: io_hdf5_write_dataset_double_3d**
*Description:* Open and write a double 3-D array dataset in thisObject.
*Code lines:* 171
*Contained by:* module io_hdf5
*Modules used:* galacticus_error

**subroutine: io_hdf5_write_dataset_double_4d**
*Description:* Open and write a double 4-D array dataset in thisObject.
*Code lines:* 171
*Contained by:* module io_hdf5
*Modules used:* galacticus_error

**subroutine: io_hdf5_write_dataset_double_5d**
*Description:* Open and write a double 5-D array dataset in thisObject.
*Code lines:* 171
*Contained by:* module io_hdf5
*Modules used:* galacticus_error

**subroutine: io_hdf5_write_dataset_integer8_1d**
*Description:* Open and write a long integer 1-D array dataset in thisObject.
*Code lines:* 167
*Contained by:* module io_hdf5
*Modules used:* galacticus_error, kind_numbers, memory_management

**subroutine: io_hdf5_write_dataset_integer_1d**
*Description:* Open and write an integer 1-D array dataset in thisObject.
*Code lines:* 159
*Contained by:* module io_hdf5
*Modules used:* galacticus_error
subroutine: io_hdf5_write_dataset_varstring_1d
Description: Open and write a varying string 1-D array dataset in thisObject.
Code lines: 15
Contained by: module io_hdf5
Modules used: string_handling

subroutine: io_hdf_assert_is_initialized
Description: Check if this module has been initialized.
Code lines: 7
Contained by: module io_hdf5
Modules used: galacticus_error

file: utility.IO.XML.F90
Description: Contains a module which implements various utility functions for extracting data from XML files.
Code lines: 319

module: io_xml
Description: Implements various utility functions for extracting data from XML files.
Code lines: 299
Contained by: file utility.IO.XML.F90
Used by: subroutine atomic_data_initialize subroutine chemical_densities_cie_file_interpolate subroutine chemical_state_cie_file_read function electron_density_cie_file_interpolate function electron_density_cie_file_logtemperature_interpolate function cooling_function_cie_file_interpolate function cooling_function_cie_file_logtemperature_interpolate subroutine galacticus_time_per_tree_file_initialize subroutine load_primus_mass_function subroutine galacticus_merger_tree_lightcone_geometry_initialize subroutine galacticus_version_output subroutine filter_response_load subroutine filereaddata subroutine radiation_igb_file_initialize subroutine imf_energy_input_rate_noninstantaneous subroutine imf_metal_yield_rate_noninstantaneous subroutine imf_recycling_rate_noninstantaneous subroutine supernovae_population_iii_hegerwoosley_initialize subroutine supernovae_type_ia_hagashima_initialize subroutine critical_overdensity_mass_scaling_wdm_initialize
subroutine halo_mass_function_tinker2008_initialize
program tests_io_xml

function: xml_array_length
Description: Return the length of an array of XML elements.
Code lines: 11
Contained by: module io_xml
Modules used: fox_dom

interface: xml_array_read
Code lines: 4
Contained by: module io_xml

subroutine: xml_array_read_one_column
Description: Read one column of data from an array of XML elements.
Code lines: 21
Contained by: module io_xml
Modules used: fox_dom memory_management

interface: xml_array_read_static
Code lines: 5
Contained by: module io_xml

subroutine: xml_array_read_static_one_column
Description: Read one column of data from an array of XML elements.
Code lines: 19
Contained by: module io_xml
Modules used: fox_dom

subroutine: xml_array_read_two_column
Description: Read two columns of data from an array of XML elements.
Code lines: 23
Contained by: module io_xml
Modules used: fox_dom memory_management

subroutine: xml_extrapolation_element_decode
Description: Extracts information from a standard XML extrapolationElement. Optionally a set of allowedMethods can be specified—if the extracted method does not match one of these an error is issued.
Code lines: 42
Contained by: module io_xml
Modules used: fox_dom galacticus_error

function: xml_get_first_element_by_tag_name
Description: Return a pointer to the first node in an XML node that matches the given tagName.
Code lines: 31
Contained by: module io_xml
Modules used: fox_dom galacticus_error
subroutine: xml_list_array_read_one_column
   Description: Read one column of data from an array of XML elements.
   Code lines: 20
   Contained by: module io_xml
   Modules used: fox_dom memory_management

subroutine: xml_list_character_array_read_static_one_column
   Description: Read one column of character data from an array of XML elements.
   Code lines: 18
   Contained by: module io_xml
   Modules used: fox_dom

subroutine: xml_list_double_array_read_static_one_column
   Description: Read one column of integer data from an array of XML elements.
   Code lines: 18
   Contained by: module io_xml
   Modules used: fox_dom

subroutine: xml_list_integer_array_read_static_one_column
   Description: Read one column of integer data from an array of XML elements.
   Code lines: 18
   Contained by: module io_xml
   Modules used: fox_dom

function: xml_path_exists
   Description: Return true if the supplied path exists in the supplied xmlElement.
   Code lines: 32
   Contained by: module io_xml
   Modules used: fox_dom

file: utility.arrays.F90
   Description: Contains a module which implements useful operations on arrays.
   Code lines: 404

module: array_utilities
   Description: Contains routines which implement useful operations on arrays.
   Code lines: 384
   Contained by: file utility.arrays.F90
   Used by: function fileconstructor subroutine cooling_radius_simple_-initialize subroutine cooling_rate_white_frenk_-initialize subroutine cooling_rate_simple_-scaling_initialize function einastoconstructor function betaprofiledefaultconstructor function ricotti2000defaultconstructor subroutine merger_tree_read_do subroutine sussingtreeindicesread
19.1. Program units

subroutine sussingtreeindicesread subroutine merger_tree_data_structure_export_irate
function table1d_is_monotonic subroutine table_id_reverse
subroutine radiation_igb_file_initialize subroutine satellite_merging_remmant_progenitor_properties_cole2000_init
subroutine satellite_merging_remmant_progenitor_properties_standard_init subroutine star Formation_timescale_disks_dynamical_time_initialize
subroutine critical_overdensity_initialize subroutine test_array_monotonicity
program test_make_ranges program test_nodes
program test_tables

interface: array_cumulate
Description: Interface to generic routines which cumulate values in an array.
Code lines: 3
Contained by: module array_utilities

function: array_cumulate_double
Description: Cumulates values in a double precision array.
Code lines: 14
Contained by: module array_utilities

interface: array_index
Description: Interface to generic routines which return a subset of an array given indices into the array.
Code lines: 6
Contained by: module array_utilities

function: array_index_double
Description: Return a subset of a double precision array given a set of indices into the array.
Code lines: 12
Contained by: module array_utilities

function: array_index_double_2d
Description: Return a subset of a 2D double precision array given a set of indices into the array.
Code lines: 28
Contained by: module array_utilities
Modules used: galacticus_error

function: array_index_integer
Description: Return a subset of an integer array given a set of indices into the array.
Code lines: 12
Contained by: module array_utilities

function: array_index_integer8
Description: Return a subset of an integer array given a set of indices into the array.
Code lines: 13
Contained by: module array_utilities
Modules used: kind_numbers
function: array_intersection_varying_string
  Code lines: 22
  Contained by: module array_utilities
  Modules used: iso_varying_string

interface: array_is_monotonic
  Description: Interface to generic routines which check if an array is monotonic.
  Code lines: 4
  Contained by: module array_utilities

function: array_is_monotonic_double
  Description: Checks if a double precision array is monotonic.
  Code lines: 82
  Contained by: module array_utilities

function: array_is_monotonic_integer8
  Description: Checks if an integer array is monotonic.
  Code lines: 83
  Contained by: module array_utilities
  Modules used: kind_numbers

interface: array_reverse
  Description: Interface to generic routines which reverse the direction of an array.
  Code lines: 5
  Contained by: module array_utilities

function: array_reverse_double
  Description: Reverses the direction of a double precision array.
  Code lines: 11
  Contained by: module array_utilities

function: array_reverse_real
  Description: Reverses the direction of a real array.
  Code lines: 11
  Contained by: module array_utilities

function: array_reverse_sizet
  Description: Reverses the direction of a real array.
  Code lines: 12
  Contained by: module array_utilities
  Modules used: iso_c_binding

subroutine: array_which
  Description: Return an array of indices for which mask is true.
  Code lines: 18
  Contained by: module array_utilities
  Modules used: galacticus_error
interface: operator(.intersection.)
  Code lines: 2
  Contained by: module array_utilities

file: utility.command_arguments.F90
  Description: Contains a module which provides an interface to read command line arguments of arbitrary type.
  Code lines: 124

module: command_arguments
  Description: Provides an interface to read command line arguments of arbitrary type.
  Code lines: 103
  Contained by: file utility.command_arguments.F90
  Used by: program bolshoi_merger_tree_file_maker    program conditional_mass_function
           program millennium_merger_tree_file_maker    program simple_merger_tree_file_maker

interface: get_argument
  Description: Generic interface to routines that read command line arguments.
  Code lines: 8
  Contained by: module command_arguments

subroutine: get_argument_character
  Description: Reads a character command line argument.
  Code lines: 8
  Contained by: module command_arguments

subroutine: get_argument_double
  Description: Reads a double command line argument.
  Code lines: 10
  Contained by: module command_arguments

subroutine: get_argument_integer
  Description: Reads a integer command line argument.
  Code lines: 10
  Contained by: module command_arguments

subroutine: get_argument_logical
  Description: Reads a logical command line argument.
  Code lines: 10
  Contained by: module command_arguments

subroutine: get_argument_real
  Description: Reads a real command line argument.
  Code lines: 10
  Contained by: module command_arguments
subroutine: get_argument_varying_string
  Description: Reads a varying string command line argument.
  Code lines: 12
  Contained by: module command_arguments
  Modules used: iso_varying_string

subroutine: get_temporary_string
  Description: Reads a command line argument into a temporary string of the correct length, and returns it as a varying string.
  Code lines: 11
  Contained by: module command_arguments
  Modules used: iso_varying_string

file: utility.date_and_time.F90
  Description: Contains a module which implements computation of formatted dates and times.
  Code lines: 87

module: dates_and_times
  Description: Implements computation of formatted dates and times.
  Code lines: 67
  Contained by: file utility.date_and_time.F90
  Used by: program xray_absorption_ism_wilms2000 subroutine galacticus_version_output subroutine merger_trees_bolshoi_process subroutine merger_trees_millennium_process subroutine merger_trees_simple_process subroutine merger_tree_write function imf_energy_input_rate_noninstantaneous function imf_metal_yield_rate_noninstantaneous function imf_recycling_rate_noninstantaneous

function: formatted_date_and_time
  Description: Return a formatted date and time.
  Code lines: 49
  Contained by: module dates_and_times
  Modules used: iso_varying_string string_handling

file: utility.files.F90
  Description: Contains a module which stores file units and finds available file units.
  Code lines: 143

module: file_utilities
  Description: Contains a function which returns an available file unit. Also stores the name of the output directory and unit numbers for various files which remain open throughout.
  Code lines: 119
  Contained by: file utility.files.F90
  Modules used: iso_c_binding iso_varying_string
  Used by: subroutine galacticus_build_output subroutine galacticus_version_output subroutine geometry_survey_window_functions_caputi_2011_ukidss_uds subroutine geometry_survey_window_functions_li_white_2009_sdss
subroutine geometry_survey_window_
  functions_martin_2010_alfalfa
subroutine sussingtreeindicesread
subroutine sussingtreeindicesread
subroutine merger_trees_millennium_
  process
subroutine merger_trees_render_dump
subroutine merger_tree_data_structure_
  read_ascii
function imf_energy_input_rate_
  noninstantaneous
function imf_metal_yield_rate_
  noninstantaneous
function imf_recycling_rate_
  noninstantaneous
function stellar_population_luminosity
subroutine excursion_sets_first_
  crossing_farahi_read_file
program test.zhao2009_flat
program test.zhao2009_open
subroutine io_hdf5_open_file

interface: count_lines_in_file
  Description: Generic interface for Count_Lines_in_File function.
  Code lines: 4
  Contained by: module file_utilities

function: count_lines_in_file_char
  Description: Returns the number of lines in the file in_file (version for character argument).
  Code lines: 27
  Contained by: module file_utilities
  Modules used: galacticus_error

function: count_lines_in_file_varstr
  Description: Returns the number of lines in the file in_file (version for varying string argument).
  Code lines: 12
  Contained by: module file_utilities

interface: file_exists
  Description: Generic interface for functions that check for a files existance.
  Code lines: 4
  Contained by: module file_utilities

function: file_exists_char
  Description: Checks for existence of file FileName (version for character argument).
  Code lines: 7
  Contained by: module file_utilities

function: file_exists_varstr
  Description: Checks for existence of file FileName (version for varying string argument).
19. Source Code Documentation

**Function:** file_lock

*Description:* Place a lock on a file.

**Subroutine:** file_unlock

*Description:* Remove a lock from a file.

**File:** utility.hashes.F90

*Description:* Contains a module which implements “hashes” (i.e. associative arrays).

**Module:** hashes

*Description:* Implements “hashes” (i.e. associative arrays).

**Subroutine:** delete_integer_scalar_ch

*Description:* Deletes entry key from thisHash.

**Subroutine:** delete_integer_scalar_vs

*Description:* Deletes entry key from Hash.

**Subroutine:** destroy_integer_scalar

*Description:* Destroys thisHash.

**Function:** exists_integer_scalar_ch

*Description:* Returns true if the specified key exists in the specified thisHash, false otherwise.

**Function:** exists_integer_scalar_vs
19.1. Program units

**Description:** Returns true if the specified key exists in the specified thisHash, false otherwise.

**Code lines:** 15

**Contained by:** module hashes

**subroutine:** initialize_integer_scalar

**Description:** Routine to initialize (or re-initialize) an integer hash.

**Code lines:** 13

**Contained by:** module hashes

**type:** integerscalarhash

**Description:** Derived type for integer hashes.

**Code lines:** 89

**Contained by:** module hashes

**function:** key_integer_scalar_i

**Description:** Returns the key of entry number index in thisHash.

**Code lines:** 12

**Contained by:** module hashes

**subroutine:** keys_integer_scalar

**Description:** Returns an array of all keys in thisHash.

**Code lines:** 13

**Contained by:** module hashes

**subroutine:** set_integer_scalar_ch

**Description:** Sets the value of key in thisHash to value.

**Code lines:** 11

**Contained by:** module hashes

**subroutine:** set_integer_scalar_vs

**Description:** Sets the value of key in thisHash to value.

**Code lines:** 68

**Contained by:** module hashes

**Modules used:**
- arrays_search
- iso_c_binding

**function:** size_integer_scalar

**Description:** Returns the number of elements in the specified Hash.

**Code lines:** 10

**Contained by:** module hashes

**function:** value_integer_scalar_ch

**Description:** Returns the value of Key in Hash.

**Code lines:** 10

**Contained by:** module hashes

**function:** value_integer_scalar_i

**Description:** Returns the value of entry number index in Hash.

**Code lines:** 11

**Contained by:** module hashes
function: value_integer_scalar_vs
Description: Returns the value of key in thisHash.
Code lines: 20
Contained by: module hashes
Modules used: arrays_search, galacticus_error, iso_c_binding

subtype: values_integer_scalar
Description: Returns an array of all values in thisHash.
Code lines: 13
Contained by: module hashes

file: utility.hashes.cryptographic.F90
Code lines: 64

module: hashes_cryptographic
Code lines: 43
Contained by: file utility.hashes.cryptographic.F90
Modules used: iso_c_binding
Used by: program test_hashes_cryptographic, module input_parameters

function: hash_md5
Code lines: 24
Contained by: module hashes_cryptographic
Modules used: iso_varying_string, string_handling

file: utility.hashes.perfect.F90
Description: Contains a module which implements a perfect hash algorithm for long integer keys.
Code lines: 307

module: hashes_perfect
Description: Implements a perfect hash algorithm for long integer keys based on methods described by Czech et al. [1997]. The specific implementation follows the general structure of that given in a Dr. Dobbs article.
Code lines: 287
Contained by: file utility.hashes.perfect.F90
Modules used: iso_c_binding, kind_numbers
Used by: program test_perfect_hashes

subtype: hash_perfect_create
Description: Create a perfect hash for a given set of keys.
Code lines: 139
Contained by: module hashes_perfect
Modules used: galacticus_error, memory_management

subtype: hash_perfect_destroy
Description: Destroy a perfect hash.
Code lines: 11
function: hash_perfect_index
Description: Return the index corresponding to a hash key.
Code lines: 14
Contained by: module hashes_perfect
Modules used: memory_management

function: hash_perfect_is_present
Description: Returns true if the hash contains the key.
Code lines: 12
Contained by: module hashes_perfect
Modules used: galacticus_error

function: hash_perfect_size
Description: Return the size of the hash table.
Code lines: 10
Contained by: module hashes_perfect
Modules used: galacticus_error

function: hash_perfect_value
Description: Returns the value for a specified key.
Code lines: 13
Contained by: module hashes_perfect
Modules used: galacticus_error

type: hashperfect
Description: A derived type which stores perfect long integer hashes.
Code lines: 54
Contained by: module hashes_perfect


type: rowstructure
Description: A row structure used in building hashes
Code lines: 4
Contained by: module hashes_perfect

file: utility.input_parameters.F90
Description: Contains a module which implements reading of parameters from an XML data file.
Code lines: 1584

module: input_parameters
Description: Implements reading of parameters from an XML data file.
Code lines: 1564
Contained by: file utility.input_parameters.F90
Modules used: fox_dom galacticus_build galacticus_error galacticus_versioning hashes_cryptographic io_hdf5
19. Source Code Documentation

```
io_xml
iso_varying_string

Used by:
program conditional_mass_function
program galacticus
program mass_function_covariance
program simple Merger_tree_file maker

subroutine accretion_haloinitialize
function nullconstructor
function simpledefaultconstructor
function nullconstructor
function simpledefaultconstructor

subroutine accretion_disks_eddington_initialize
subroutine accretion_diskspectralinitialize
function filedefaultconstructor

subroutine atomic_cross_section_-
ionization_photo_initialize
subroutine gaunt_factor_initialize

subroutine atomic_rate_ionization_-
collisional_initialize
subroutine atomic_rate_recombination_-
radiative_initialize
subroutine black_hole_binary_initial_-
radii_spheroid_size_initialize
function black_hole_binary_separation_-
growth_rate
subroutine black_hole_binary_merger

subroutine chemical_hydrogen_rates_-
initialize
subroutine chemical_state_initialize

subroutine cooling_cold_mode_infall_-
rate_initialize
subroutine cooling_function_cie_file_-
initialize
subroutine cooling_radius_initialize
subroutine cooling_rate_initialize

iso_c_binding
```

program tests_excursion_sets
program halo_mass_functions
program power_spectra
subroutine accretion_halos_output_-
initialize
function coldmodedefaultconstructor
function singleconstructor
function coldmodedefaultconstructor
function singleconstructor
subroutine accretion_disks_adaf_get_-
parameters
subroutine accretion_disks_initialize
function filedefaultconstructor

subroutine accretion_disks_switched_-
initialize
subroutine atomic_ionization_-
potential_initialize
subroutine collisional_excitation_-
rate_initialize
subroutine dielectronic_recombination_-
ratebination_rate_initialize
function black_hole_binary_initial_-
radius
function black_hole_binary_recoil_-
velocity
subroutine black_hole_binary_-
separation_growth_rate_standard_init
subroutine black_hole_binary_-
separation_growth_rate_standard_init
subroutine chemical_reaction_rates_-
initialize
subroutine chemical_state_cie_file_-
initialize
subroutine cooling_cold_mode_infall_-
output_initialize
subroutine cooling_cold_mode_infall_-
rate_dynamical_time_initialize
subroutine cooling_function_initialize

subroutine cooling_radius_output_-
initialize
subroutine cooling_rate_output_-
initialize
subroutine cooling_rate_white_frenk_initialize
subroutine cooling_rate_simple_initialize
subroutine cooling_time_initialize
subroutine freefall_radius_initialize
subroutine infall_radius_initialize
subroutine cooling_specific_am_constant_rotation_initialize
subroutine cooling_time_available_wf_initialize
subroutine cosmologyfunctionsinitialize

function matterdarkenergymodeldefaultconstructor
subroutine cosmologyparametersinitialize
function cosmologyparameterssimplemodeldefaultconstructor
subroutine dark_matter_mass_accretion_wechsler2002_initialize
subroutine darkmatterhaloscaleinitialize
function halo_spin_distribution_sample

subroutine halo_spin_distribution_lognormal_initialize
subroutine darkmatterprofileinitialize
function isothermalconstructor
function nfwdefaultconstructor
function isothermalconstructor
function diemerkravtsov2014defaultconstructor
subroutine darkmatterprofileconcentrationinitialize
diemerkravtsov2014defaultconstructor

function einastodefaultconstructor
function isothermaldefaultconstructor
function nfwdefaultconstructor
function isothermaldefaultconstructor
function duttonmaccio2014defaultconstructor
duttonmaccio2014defaultconstructor

function nfw1996defaultconstructor
function prada2011defaultconstructor
function zhao2009defaultconstructor
function prada2011defaultconstructor
function zhao2009defaultconstructor
subroutine dark_matter_shapes_initialize
subroutine galactic_dynamics_bar_instabilities_eln_initialize
subroutine galactic_dynamics_bar_instability_initialize
subroutine galactic_structure_radii_adiabatic_initialize
subroutine galactic_structure_radii_initial_initialize
subroutine galactic_structure_radii_simple_initialize
subroutine galacticus_time_per_tree_file_initialize
subroutine meta_tree_timing_initialize
subroutine galacticus_output_open_file

subroutine alfalfa_hi_mass_function_z0_00_initialize
subroutine galacticus_merger_tree_output
subroutine galacticus_merger_tree_descendents_initialize
subroutine galacticus_merger_tree_lightcone_geometry_initialize
subroutine galacticus_merger_tree_output_filter_luminosity_initialize
subroutine galacticus_merger_tree_output_stllr_mass_morphlgy_initialize
subroutine galacticus_merger_tree_output_half_light_initialize
subroutine galacticus_output_halo_model_initialize
subroutine galacticus_output_tree_mass_profile_initialize
subroutine galacticus_output_tree_redshifts_initialize
subroutine galacticus_output_tree_satellite_extremum_initialize
subroutine galacticus_output_tree_satellite_status_initialize
subroutine star_formation_histories_metallicity_split_initialize
subroutine galacticus_output_tree_velocity_dispersion_initialize
subroutine output_times_initialize
function galacticus_task_start

subroutine galactic_dynamics_bar_instabilities_eln_tidal_initialize
subroutine galactic_structure_radii_solve
subroutine galactic_structure_radii_fixed_initialize
subroutine galactic_structure_initial_radii_adiabatic_initialize
subroutine galacticus_time_per_tree_initial_initialize
subroutine galacticus_meta_evolver_profile
subroutine galacticus_output_close_file
subroutine galacticus_output_analysis_mass_dpndnt_sz_dstrbtins
subroutine galacticus_output_analysis_mass_functions
subroutine galacticus_output_tree_density_contrast_initialize
subroutine galacticus_merger_tree_output_filter_initialize
subroutine merger_tree_prune_lightcone

subroutine galacticus_merger_tree_output_filter_luminosity_initialize
subroutine galacticus_merger_tree_output_filter_stellar_mass_initialize
subroutine galacticus_merger_tree_final_descendents_initialize
subroutine galacticus_merger_tree_half_mass_initialize
subroutine galacticus_merger_tree_main_branch_initialize
subroutine galacticus_output_most_massive_progenitor_initialize
subroutine galacticus_output_tree_rotation_curve_initialize
subroutine galacticus_output_tree_satellite_host_initialize
subroutine galacticus_output_star_formation_histories_initialize
subroutine galacticus_output_tree_indices_initialize
subroutine galacticus_output_tree_virial_initialize
subroutine state_initialize
function galacticus_task_evolve_tree
subroutine geometry_surveys_caputi_2011_ukidsd_uds_initialize
subroutine geometry_surveys_li_white_2009_sdss_initialize
subroutine halo_mass_function_compute
subroutine conditional_mass_functions_initialize
subroutine hot_halo_cold_mode_density_ciso_corer_vf_initialize
function betaprofileconstructor
function betaprofiledefaultconstructor
subroutine hot_halo_cold_mode_density_ciso_corer_VF_initialize
function growingconstructor
function growingdefaultconstructor
function virialfractionconstructor
function virialfractiondefaultconstructor
function hot_halo_ram_pressure_force
function hot_halo_ram_pressure_stripping_radius
function hot_halo_ram_pressure_stripping_timescale
function filedefaultconstructor
function internaldefaultconstructor
function recfastconstructor
function simpledefaultconstructor
subroutine intergalactic_medium_state_internal_initialize
subroutine tree_branching_initialize
subroutine modified_press_schechter_branching_initialize
subroutine merger_tree_branching_modifiers_parkinson_initialize
function merger_tree_create
function merger_tree_correlation
function merger_tree_correlation_radius
function merger_tree_conditionals_mass_function
function merger_tree_conditionals_mass_function_radius
subroutine merger_tree_build_cole2000_initialize
subroutine merger_tree_build_mass_resolution_fixed_initialize
subroutine merger_tree_build_mass_resolution_scaled_initialize
subroutine merger_tree_construct_fully_specified_initialize
subroutine merger_trees_mass_function_sampling_gaussian_initialize
subroutine merger_trees_mass_function_sampling_power_law_initialize
subroutine assign_scale_radii
subroutine time_until_merging_subresolution
subroutine merger_tree_smooth_accretion_initialize
subroutine merger_tree_read_initialize
subroutine time_until_merging_subresolution
function galacticusdefaultconstructor
function sussingdefaultconstructor
function sussingdefaultconstructor
function galacticusdefaultconstructor
subroutine merger_tree_timestep_history
subroutine merger_tree_timestep_simple
subroutine merger_trees_bolshoi_process
subroutine chemical_abundances_initialize
subroutine galacticus_nodes_initialize
function node_component_black_hole_standard_seed_mass
subroutine tree_node_create_initialize
subroutine node_component_black_hole_standard_initialize
subroutine node_component_dark_matter_profile_scale_shape_initialize
subroutine node_component_disk_profile_scale_shape_initialize
subroutine node_component_disk_very_simple_initialize
subroutine node_component_formation_times_cole2000_initialize
subroutine merger_trees_mass_function_sampling_halo_mf_initialize
subroutine merger_trees_mass_function_sampling_star_mf_initialize
subroutine merger_trees_mass_function_sampling_stellar_mf_initialize
subroutine merger_tree_evolve_to
subroutine tree_node_evolve_initialize
subroutine merger_tree_timestep_record_evolution
subroutine merger_tree_timestep_subresolution
subroutine merger_tree_timestep_satellite
subroutine merger_trees_mass_accretion_history_output
subroutine merger_tree_structure_output
subroutine merger_tree_prune_hierarchy
subroutine merger_tree_prune_branches
subroutine merger_tree_write
subroutine chemical_abundances_initialize
function node_component_black_hole_simple_seed_mass
function node_component_black_hole_standard_seed_mass
function node_component_black_hole_simple_seed_mass
function node_component_black_hole_standard_seed_spin
function node_component_black_hole_simple_initialize
function node_component_black_hole_standard_initialize
subroutine node_component_dynamics_statistics_bars_timestep
subroutine node_component_disk_exponential_initialize
subroutine node_component_disk_very_simple_initialize
subroutine node_component_dynamics_statistics_bars_timestep
subroutine node_component_hot_halo_cold_mode_initialize
19.1. Program units

subroutine node_component_hot_halo_standard_initialize
subroutine node_component_mass_flow_statistics_standard_initialize
subroutine node_component_merging_statistics_recent_initialize
subroutine node_component_satellite_orbiting_initialize
subroutine node_component_satellite_varying_initialize
subroutine node_component_satellite_varying_simple_initialize
subroutine node_component_satellite_orbiting_initialize
subroutine node_component_satellite_standard_initialize
subroutine node_component_satellite씁니다
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_satellite_varying_standard_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_satellite_varying_standard_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_standard_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standard_initialize
subroutine node_component_mass_flow_statistics_recent_initialize
subroutine node_component_satellite_orbiting_standardize
subroutine virial_orbital_parameters_fixed_initialize
subroutine satellites_tidal_fields_null_initialize
subroutine satellite_tidal_heating_rate_initialize
subroutine satellite_tidal_stripping_initialize
subroutine star_formation_imf_initialize_baugh2005topheavy
module star_formation_imf
subroutine star_formation_imf_initialize_kroupa
subroutine star_formation_imf_initialize_salpeter
subroutine star_formation_imf_initialize_piecewisepowerlaw
subroutine imf_select_fixed_initialize
subroutine star_formation_feedback_disks_initialize
subroutine star_formation_feedback_disks_fixed_initialize
subroutine star_formation_feedback_disks_halo_scaling_initialize
subroutine star_formation_feedback_spheroids_initialize
subroutine star_formation_expulsive_feedback_disks_initialize
subroutine star_formation_expulsive_feedback_spheroids_initialize
subroutine star_formation_rate_surface_density_disks_br_initialize
subroutine star_formation_rate_surface_density_disks_kmt09_initialize
subroutine star_formation_rate_surface_density_disks_exschmidt_initialize
subroutine star_formation_timescale_disks_initialize
subroutine star_formation_timescale_disks_fixed_initialize
subroutine star_formation_timescale_disks_halo_scaling_initialize
subroutine star_formation_timescale_spheroids_initialize
subroutine star_formation_timescale_spheroids_dynamical_time_initialize
function satellite_tidal_field
subroutine satellites_tidal_fields_spherical_symmetry_initialize
subroutine satellite_tidal_heating_rate_gnedin_initialize
subroutine satellite_tidal_stripping_rate_zentner2005_initialize
subroutine star_formation_imf_initialize_chabrier
subroutine star_formation_imf_initialize_kennicutt
subroutine star_formation_imf_initialize_millerscalo
subroutine star_formation_imf_initialize_salpeter
subroutine star_formation_imf_initialize_piecewisepowerlaw
subroutine star_formation_feedback_disks_creasey2012_initialize
subroutine star_formation_feedback_disks_power_law_initialize
subroutine star_formation_feedback_spheroids_power_law_initialize
subroutine star_formation_expulsive_feedback_disks_sw_initialize
subroutine star_formation_expulsive_feedback_spheroids_sw_initialize
subroutine star_formation_rate_surface_density_disks_ks_initialize
subroutine star_formation_timescale_disks_dynamical_time_initialize
subroutine star_formation_timescale_disks_fixed_initialize
subroutine mass_function_covariance_matrix
subroutine stellar_astrophysics_initialize
subroutine stellar_feedback_standard_initialize
subroutine supernovae_population_iii_initialize
subroutine stellar_tracks_initialize
subroutine stellar_winds_initialize
subroutine stellar_population_properties_rates_initialize
subroutine stellar_population_spectrum_initialize
function recentdefaultconstructor
function stellar_population_spectrum_postprocess_index
subroutine cosmological_mass_variance_filtered_power_spectrum_initialize
subroutine critical_overdensity_mass_filtering_initialize
subroutine excursion_sets_barriers_quadratic_initialize
subroutine excursion_sets_first_crossings_initialize
subroutine dark_matter_halo_bias_initialize
subroutine halo_mass_function_modifier_simple_systematic
subroutine initialize_cosmological_mass_variance
subroutine power_spectrum_nonlinear_cosmicemu_initialize
subroutine power_spectrum_initialize
subroutine power_spectrum_window_functions_initialize
subroutine power_spectrum_window_functions_th_kss_hybrid_initialize
subroutine transfer_function_bbks_initialize
subroutine stellar_feedback_initialize
subroutine stellar_astrophysics_file_initialize
subroutine supernovae_type_ia_initialize
subroutine stellar_tracks_initialize_file
function stellar_population_luminosity
subroutine stellar_population_properties_noninstantaneous_initialize
subroutine stellar_population_spectra_file_initialize_imf
subroutine spectrum_postprocessor_initialize
function recentdefaultconstructor
subroutine critical_overdensity_initialize
subroutine critical_overdensity_mass_scaling_wdm_initialize
subroutine excursion_sets_barriers_linear_initialize
subroutine excursion_sets_barriers_remap_scale_initialize
subroutine excursion_sets_first_crossing_farahi_initialize
subroutine halo_mass_function_initialize
subroutine linear_growth_initialize
function power_spectrum_nonlinear_cosmicemu
subroutine power_spectrum_nonlinear_cosmicemu_initialize
subroutine primordial_power_spectrum_power_law_initialize
subroutine power_spectrum_window_functions_sharp_kspace_initialize
subroutine make_table
subroutine transfer_function_camb_make
subroutine transfer_function_eisenstein_hu_initialize
definition

subroutine transfer_function_file_initialize
definition

function friendsoffriendsconstructor

function sphericalcollapsematterdefaultconstructor
definition

function sphericalcollapsematterlambda_defaultconstructor
definition

subroutine virialdensitycontrastinitialize
definition

function friendsoffriendsconstructor

function sphericalcollapsematterdefaultconstructor
definition

program test_nfsw96_concentration_dark_energy
definition

program test_zhao2009_flat

program test_zhao2009_open

program tests_bug745815

program tests_cosmic_age

program tests_kepler_orbits

program tests_linear_growth_cosmological_constant

program tests_linear_growth_open

program tests_power_spectrum

program tests_spherical_spectrum_dark_energy_ed

program tests_spherical_collapse_dark_energy_omega_zero_point_six

program tests_spherical_collapse_dark_energy_omega_zero_point_eight

program tests_spherical_collapse_dark_energy_omega_two_thirds

program tests_spherical_collapse_dark_energy_open

program tests_spherical_collapse_open

subroutine tidal_stripping_mass_loss_rate_disks_null_init
function tidal_stripping_mass_loss_rate_spheroid

subroutine tidal_stripping_mass_loss_rate_spheroids_simple_init

function tidal_stripping_mass_loss_rate_disk

subroutine tidal_stripping_mass_loss_rate_disks_simple_init

subroutine tidal_stripping_mass_loss_rate_spheroids_null_init

subroutine close_parameters_group

Code lines:  7
Contained by: module input_parameters

interface: get_input_parameter
function: get_input_parameter_array_size
Description: Get the number of elements in the parameter specified by parameter name is specified by parameterName.
Code lines: 36
Contained by: module input_parameters
Modules used: string_handling

subroutine: get_input_parameter_char
Description: Read a varying_string parameter from the parameter file. The parameter name is specified by parameterName and its value is returned in parameterValue. If no parameter file has been opened by Input_Parameters_File_Open or no matching parameter is found, the default value (if any) given by defaultValue is returned. (If no default value is present an error occurs instead.)
Code lines: 60
Contained by: module input_parameters
Modules used: string_handling

subroutine: get_input_parameter_char_array
Description: Read a varying_string parameter from the parameter file. The parameter name is specified by parameterName and its value is returned in parameterValue. If no parameter file has been opened by Input_Parameters_File_Open or no matching parameter is found, the default value (if any) given by defaultValue is returned. (If no default value is present an error occurs instead.)
Code lines: 68
Contained by: module input_parameters
Modules used: string_handling

subroutine: get_input_parameter_double
Description: Read a double precision parameter from the parameter file. The parameter name is specified by parameterName and its value is returned in parameterValue. If no parameter file has been opened by Input_Parameters_File_Open or no matching parameter is found, the default value (if any) given by defaultValue is returned. (If no default value is present an error occurs instead.)
Code lines: 59
Contained by: module input_parameters

subroutine: get_input_parameter_double_array
Description: Read a double precision parameter from the parameter file. The parameter name is specified by parameterName and its value is returned in parameterValue. If no parameter file has been opened by Input_Parameters_File_Open or no matching parameter is found, the default value (if any) given by defaultValue is returned. (If no default value is present an error occurs instead.)
Code lines: 60
Contained by: module input_parameters

subroutine: get_input_parameter_double_c
Description: C-bound wrapper function for getting double precision parameter values.
subroutine: get_input_parameter_integer

Description: Read a integer parameter from the parameter file. The parameter name is specified by parameterName and its value is returned in parameterValue. If no parameter file has been opened by Input_Parameters_File_Open or no matching parameter is found, the default value (if any) given by defaultValue is returned. (If no default value is present an error occurs instead.)

subroutine: get_input_parameter_integer_array

Description: Read an integer parameter from the parameter file. The parameter name is specified by parameterName and its value is returned in parameterValue. If no parameter file has been opened by Input_Parameters_File_Open or no matching parameter is found, the default value (if any) given by defaultValue is returned. (If no default value is present an error occurs instead.)

subroutine: get_input_parameter_integer_c

Description: C-bound wrapper function for getting integer parameter values.

subroutine: get_input_parameter_integer_long

Description: Read a long integer parameter from the parameter file. The parameter name is specified by parameterName and its value is returned in parameterValue. If no parameter file has been opened by Input_Parameters_File_Open or no matching parameter is found, the default value (if any) given by defaultValue is returned. (If no default value is present an error occurs instead.)

subroutine: get_input_parameter_integer_long_array

Description: Read a long integer parameter from the parameter file. The parameter name is specified by parameterName and its value is returned in parameterValue. If no parameter file has been opened by Input_Parameters_File_Open or no matching parameter is found, the default value (if any) given by defaultValue is returned. (If no default value is present an error occurs instead.)

subroutine: get_input_parameter_logical
19.1. Program units

**Description:** Read a logical parameter from the parameter file. The parameter name is specified by parameterName and its value is returned in parameterValue. If no parameter file has been opened by Input_Parameters_File_Open or no matching parameter is found, the default value (if any) given by defaultValue is returned. (If no default value is present an error occurs instead.)

**Code lines:** 67
**Contained by:** module input_parameters

**subroutine:** get_input_parameter_logical_array

**Description:** Read an logical parameter from the parameter file. The parameter name is specified by parameterName and its value is returned in parameterValue. If no parameter file has been opened by Input_Parameters_File_Open or no matching parameter is found, the default value (if any) given by defaultValue is returned. (If no default value is present an error occurs instead.)

**Code lines:** 66
**Contained by:** module input_parameters

**subroutine:** get_input_parameter_varstring

**Description:** Read a varying_string parameter from the parameter file. The parameter name is specified by parameterName and its value is returned in parameterValue. If no parameter file has been opened by Input_Parameters_File_Open or no matching parameter is found, the default value (if any) given by defaultValue is returned. (If no default value is present an error occurs instead.)

**Code lines:** 60
**Contained by:** module input_parameters

**subroutine:** get_input_parameter_varstring_array

**Description:** Read a varying_string parameter from the parameter file. The parameter name is specified by parameterName and its value is returned in parameterValue. If no parameter file has been opened by Input_Parameters_File_Open or no matching parameter is found, the default value (if any) given by defaultValue is returned. (If no default value is present an error occurs instead.)

**Code lines:** 68
**Contained by:** module input_parameters

**Modules used:** string_handling

**function:** input_parameter_is_present

**Description:** Return true if parameterName is present in the input file.

**Code lines:** 21
**Contained by:** module input_parameters

**subroutine:** input_parameters_file_close

**Description:** Close the parameter file (actually just destroy the internal record of it and clean up memory).

**Code lines:** 7
**Contained by:** module input_parameters

**subroutine:** input_parameters_file_open

**Description:** Open an XML data file containing parameter values and parse it. The file should be structured as follows:

```xml
<parameters>
```
19. Source Code Documentation

<parameter>
    <name>parameter1Name</name>
    <value>parameter1Value</value>
</parameter>

<parameter>
    <name>parameter1Name</name>
    <value>parameter1Value</value>
</parameter>

Code lines: 136
Contained by: module input_parameters
Modules used: file_utilities galacticus_display
galacticus_input_paths regular_expressions
string_handling

type: inputparameterlist
Description: A class to hold lists of parameters (and values) prior to output.
Code lines: 23
Contained by: module input_parameters

subroutine: inputparameterlistadd
Description: Add a parameter to a list of input parameters to an XML document.
Code lines: 25
Contained by: module input_parameters

subroutine: inputparameterlistdestructor
Description: Destroy an inputParameterList object.
Code lines: 8
Contained by: module input_parameters

subroutine: inputparameterlistoutputtoxml
Description: Write a list of input parameters to an XML document.
Code lines: 13
Contained by: module input_parameters
Modules used: fox_wxml

subroutine: make_parameters_group
Description: Create a group in the GALACTICUS output file in which to store parameters.
Code lines: 9
Contained by: module input_parameters

function: star_formation_imf_label
Code lines: 162
Contained by: module input_parameters
function: stellar_population_luminosities_label
Code lines: 133
Contained by: module input_parameters

function: transfer_function_camb_label
Code lines: 74
Contained by: module input_parameters

subroutine: write_parameter_xml
Description: Add a parameter to the specified XML file.
Code lines: 16
Contained by: module input_parameters
Modules used: fox_wxml

file: utility.kind_numbers.F90
Description: Contains a module which defines various kind types.
Code lines: 32

module: kind_numbers
Description: Defines various kind types.
Code lines: 13
Contained by: file utility.kind_numbers.F90
Used by: subroutine accretion_halos_hot_halo_output
subroutine cooling_radius_hot_halo_output
module cooling_radii_simple
module cooling_specific-angular-momenta_constant_rotation
file dark_matter_halos.scales.virial-density_contrast.F90
module dark_matter_profiles
galacticus_output_merge_tree
subroutine accretion_halos_hot_halo_output
subroutine cooling_cold_mode_infall_output
module cooling_radii_isothermal
subroutine cooling_rate_hot_halo_output
module dark_matter_halo_scales
dark_matter_profiles.F90
subroutine cooling_specific-angular-momenta_constant_rotation
file dark_matter_profiles.Einasto.F90
dark_matter_profiles.NFW.F90
galacticus_output_merge_tree
subroutine galacticus_output_merge_tree_density_contrast
subroutine galacticus_output_merge_tree_lightcone
subroutine galacticus_output_merge_tree_final_descendents
subroutine galacticus_output_merge_tree_half_mass
subroutine galacticus_output_halo_model
subroutine galacticus_output_tree_main_branch
subroutine galacticus_output_most_massive_progenitor
subroutine galacticus_output_tree_links
subroutine galacticus_output_tree_mass_profile
subroutine galacticus_output_tree_redshifts
19.1. Program units

function radiation_intergalactic_background_internal_update
module star_formation_rate_surface_density_disks_br
module star_formation_rate_surface_density_disks_kt
module star_formation_rate_surface_density_disks_exschmidt
module star_formation_timescale_disks_halo_scaling
function excursion_sets_first_crossing_probability_farahi
subroutine excursion_sets_first_crossing_rate_tabulate_farahi
program test_array_monotonicity
program test_perfect_hashes
program test_search
program test_sort
program tests_tree_branch_destroy

subroutine io_hdf5_read_attribute_integer8_1d_array
subroutine io_hdf5_read_attribute_integer8_1d_array_static
subroutine io_hdf5_read_dataset_integer8_1d_array
subroutine io_hdf5_write_attribute_integer8_1d
subroutine io_hdf5_write_data_set_integer8_1d
function array_index_integer8
function array_is_monotonic_integer8
subroutine get_input_parameter_integer_long
module memory_management
subroutine assert_integer8_1d_array

function imf_metal_yield_rate_noninstantaneous
module star_formation_rate_surface_density_disks_kmt09
module star_formation_rate_surface_density_disks_exschmidt
function excursion_sets_barrier_effective
subroutine excursion_sets_first_crossing_rate_tabulate_farahi
program tests_io_hdf5
program tests_bug745815
program test_search
program test_string_utilities
subroutine test_string_utilities
subroutine assert_integer8_scalar
subroutine io_hdf5_read_attribute_integer8_scalar
subroutine io_hdf5_write_attribute_integer8_scalar
function array_index_integer8
module hashes_perfect
subroutine get_input_parameter_integer_long
module memory_management
subroutine assert_integer8_scalar

file: utility.memory_management.F90

Description: Contains a module for storing and reporting memory usage by the code.
Code lines: 1742

module: memory_management
**Description:** Routines and data type for storing and reporting on memory usage. Also contains routines for allocating and deallocating arrays with automatic error checking and deallocation at program termination and memory usage reporting. Contains interface and type definitions for memory management routines along with storage space for pointers and sizes. This file was created automatically by tfamily Make_Memory_Usage_Routines.pl Contains memory management subroutines. This file was created automatically by tfamily Make_Memory_Usage_Routines.pl

**Code lines:** 1722

**Contained by:** file `utility.memory_management.F90`

**Modules used:**
- `galacticus_error`
- `kind_numbers`
- `iso_c_binding`

**Used by:**
- program `bolshoi_merger_tree_file_maker`
- program `tests_excursion_sets`
- program `halo_mass_functions`
- program `millennium_merger_tree_file_maker`
- program `simple_merger_tree_file_maker`
- subroutine `filedestructor`
- subroutine `atomic_data_initialize`
19.1. Program units

subroutine chemical_reaction_rates_initialize
subroutine chemical_state_cie_file_read
subroutine cooling_function_cie_file_read
subroutine cooling_function_initialize
subroutine matterdarkenergymakeexpansionfactortable
subroutine matterlambdamakedistancetable
subroutine matterlambdamakeexpansionfactortable
subroutine matterlambdamakeexpansionfactortable
subroutine matterlambdastaterestore
subroutine matterdarkenergymakeexpansionfactortable
subroutine einastoenergytablemake
subroutine einastofreefalltabulate
subroutine einastoenergytablemake
subroutine einastofreefalltabulate
subroutine solve_for_radius
subroutine meta_tree_timing_post_evolve
subroutine galacticus_output_analysis_mass_dpndnt_sz_dstrbtins
subroutine galacticus_output_analysis_mass_functions
subroutine allocate_buffers
subroutine double_buffer_extend
subroutine galacticus_output_tree_density_contrast_initialize
subroutine galacticus_output_tree_density_contrast_initialize
subroutine galacticus_output_tree_density_contrast_initialize
subroutine galacticus_output_tree_density_contrast_initialize
subroutine galacticus_output_tree_density_contrast_initialize
subroutine star_formation_histories_scales_metallicity_split
function galacticus_task_evolve_tree
subroutine geometry_survey_window_functions_caputi_2011_ukidss_uds
subroutine geometry_survey_window_functions_li_white_2009_sdss
subroutine halo_mass_function_compute
subroutine internaldensityhelium1set
subroutine internaldensityhelium2set
subroutine internaldensityhelium3set
subroutine internaldensityhydrogen2set
subroutine internaltemperatureset
subroutine internaldensityhelium1set
subroutine internaldensityhelium3set
subroutine internaldensityhydrogen2set
subroutine intergalactic_medium_state_internal_initialize
function merger_tree_create
subroutine merger_tree_construct_fully_specified
subroutine create_node_array
subroutine destroy_node_indices
subroutine merger_tree_readInitialize
subroutine sussingDestructor
subroutine sussingOpen
subroutine galacticusImport
subroutine merger_tree_timestep_history
subroutine merger_tree_mass_accretion_history_output
subroutine merger_tree_regrid_time
subroutine abundancesAllocate_elemental_values
subroutine abundancesInitialize
subroutine chemical_abundances_initialize
subroutine chemicals_read_raw
subroutine history_create
subroutine history_long_integer_clone
subroutine history_long_integer_destroy
subroutine history_long_integer_trim
subroutine sersic_profile_tabulate
subroutine merger_tree_data_structure_add_metadata
subroutine merger_tree_data_structure_export_irate
subroutine internalmassfilteringset
subroutine internaltimeset
subroutine internaldensityhelium2set
subroutine internaldensityhydrogen1set
subroutine internalmassfilteringset
subroutine internaltimeset
subroutine merger_tree_conditional_mass_function
subroutine build_child_and_sibling_links
subroutine create_node_indices
subroutine merger_tree_read_do
subroutine galacticusImport
subroutine sussingImport
subroutine sussingtreeindicesread
subroutine tree_node_evolve
subroutine merger_tree_timestep_record_evolution
subroutine merger_tree_structure_output
subroutine interpolate_2d_irregular_array
subroutine abundances_destroy
subroutine chemical_abundances_allocate_values
subroutine chemicals_abundances_destroy
subroutine history_clone
subroutine history_destroy
subroutine history_long_integer_create
subroutine history_long_integer_read_raw
subroutine history_read_raw
subroutine history_trim
subroutine merger_tree_data_construct_particle_indices
subroutine merger_tree_data_structure_export_galacticus
subroutine merger_tree_data_structure_read_ascii
subroutine merger_tree_data_structure_read_particles_ascii
subroutine merger_tree_data_structure_set_particle_property_column
subroutine merger_tree_data_structure_set_property_column
subroutine merger_tree_data_structure_set_property_double
subroutine merger_tree_data_structure_set_tree_indices
module galacticus_nodes
subroutine agestatisticsstandarddiskintegratedsfrset
subroutine agestatisticsstandarddisktimeweightedintegratedsfrset
subroutine agestatisticsstandardspheroidintegratedsfrset
subroutine agestatisticsstandardspheroidtimeweightedintegratedsfrset
subroutine basicnonevolvingmassset
subroutine basicnonevolvingtimelastisolatedset
subroutine basicstandardmassset
subroutine basicstandardtimelastisolatedset
subroutine basicstandardtimeset
subroutine blackholesimplemassset
subroutine blackholestandardmassset
subroutine blackholestandardtimeset
subroutine blackholestandardradialpositionset
subroutine blackholestandardtripleinteractiontimeset
subroutine darkmatterprofilescalescaleset
subroutine darkmatterprofilescalescalegrowthrateset
subroutine deserializefromarrayrates
subroutine deserializefromarrayscales
subroutine deserializefromarrayvalues
subroutine diskexponentialabundanceststellarset
subroutine diskexponentialisinitializedset
subroutine diskexponentialmassstellarset
subroutine diskexponentialradiusset
subroutine diskexponentialstarformationhistoryset
subroutine diskverysimplemassstellarset
subroutine diskverysimplemassgasset
subroutine dynamicsstatisticsbarsadiabaticratioset
subroutine dynamicsstatisticsbarsbarinstabilitytimescaleset
subroutine galacticus_nodes_initialize
subroutine hosthistorystandardhostmassmaximumset
subroutine hothalocoldmodeabundancescoldset
subroutine hothalocoldmodeangularmomentumcoldset
subroutine hothalocoldmodemasscoldset
subroutine hothalooutflowtrackingtrackedoutflowabundanceset
subroutine hothalooutflowtrackingtrackedoutflowmassset
subroutine hothalostandardabundanceset
subroutine hothalostandardangularmomentumset
subroutine hothalostandardmassset
subroutine hothalostandarddisinitializedset
subroutine hothalostandardouterradiusset
subroutine hothalostandardoutflowedabundanceset
subroutine hothalostandardoutflowedangularmomentumset
subroutine hothalostandardoutflowedmassset
subroutine hothalostandardstrippedabundanceset
subroutine hothalostandardstrippedmassset
subroutine hothalostandardunaccretedmassset
subroutine indicesstandardbranchtipset
subroutine interoutputstandarddiskstarformationrateset
subroutine interoutputstandardspheroidstarformationrateset
subroutine massflowstatisticsstandardcooledmassset
subroutine mergingstatisticsrecentrecentmajormergercountset
subroutine mergingstatisticsstandardgalaxymajormergertimeset
subroutine mergingstatisticsstandardnodeformationtimeset
subroutine mergingstatisticsstandardnodehierarchylevelset
subroutine mergingstatisticsstandardnodemajormergertimeset
subroutine nbodystandardparticlecountset
subroutine nbodystandardvelocitydispersionset
subroutine nbodystandardvelocitymaximumset
subroutine node_component_-agestatisticsnull_builder
subroutine node_component_-agestatisticsnull_destroy
subroutine node_component_-agestatisticsnull_initializor
subroutine node_component_-agestatisticsnull_read_raw
subroutine node_component_-agestatisticsstandard_builder
subroutine node_component_-agestatisticsstandard_destroy
subroutine node_component_-agestatisticsstandard_initializor
subroutine node_component_-agestatisticsstandard_read_raw
subroutine node_component_-basicnonevolving_builder
subroutine node_component_-basicnonevolving_destroy
subroutine node_component_basicnonevolving_initializor
subroutine node_component_basicnull_builder
subroutine node_component_basicnull_initializor
subroutine node_component_basicstandard_builder
subroutine node_component_basicstandard_destroy
subroutine node_component_basicstandard_initializor
subroutine node_component_basicstandardtracking_builder
subroutine node_component_basicstandardtracking_initializor
subroutine node_component_blackholenull_builder
subroutine node_component_blackholenull_destroy
subroutine node_component_blackholenull_initializor
subroutine node_component_blackholesimple_builder
subroutine node_component_blackholesimple_destroy
subroutine node_component_blackholesimple_initializor
subroutine node_component_blackholestandard_builder
subroutine node_component_blackholestandard_destroy
subroutine node_component_blackholestandard_initializor
subroutine node_component_darkmatterprofilenull_builder
subroutine node_component_darkmatterprofilenull_destroy
subroutine node_component_darkmatterprofilenull_initializor
subroutine node_component_darkmatterprofilescale_builder
subroutine node_component_darkmatterprofilescale_destroy
subroutine node_component_darkmatterprofilescale_initializor
subroutine node_component_darkmatterprofilescalepreset_builder
subroutine node_component_darkmatterprofilescalepreset_destroy
subroutine node_component_darkmatterprofilescalepreset_initializor
subroutine node_component_darkmatterprofilescaleshape_builder
subroutine node_component_darkmatterprofilescaleshape_destroy
subroutine node_component_darkmatterprofilescaleshape_initializor
subroutine node_component_diskexponential_builder
subroutine node_component_diskexponential_destroy
subroutine node_component_diskexponential_initializor
subroutine node_component_diskexponential_read_raw

subroutine node_component_disknull_builder
subroutine node_component_disknull_destroy
subroutine node_component_disknull_initializor
subroutine node_component_disknull_read_raw

subroutine node_component_diskverysimple_builder
subroutine node_component_diskverysimple_destroy
subroutine node_component_diskverysimple_initializor
subroutine node_component_diskverysimple_read_raw

subroutine node_component_dynamics_statistics_bars_record
subroutine node_component_dynamics_statisticsbars_destroy
subroutine node_component_dynamics_statisticsbars_initializor
subroutine node_component_dynamics_statisticsbars_read_raw
subroutine node_component_dynamics_statisticsnull_destroy
subroutine node_component_dynamics_statisticsnull_initializor
subroutine node_component_dynamics_statisticsnull_read_raw

subroutine node_component_formationtimecole2000_builder
subroutine node_component_formationtimecole2000_destroy
subroutine node_component_formationtimecole2000_initializor
subroutine node_component_formationtimecole2000_read_raw
subroutine node_component_formationtimenull_destroy
subroutine node_component_formationtimenull_initializor
subroutine node_component_formationtimenull_read_raw

subroutine node_component_hosthistorynull_destroy
subroutine node_component_hosthistorynull_initializor
subroutine node_component_hosthistorynull_read_raw
subroutine node_component_hosthistorystandard_destroy
subroutine node_component_hosthistorystandard_initializor
subroutine node_component_hosthistorystandard_read_raw

subroutine node_component_hothalocoldmode_destroy
subroutine node_component_hothalocoldmode_initializor
subroutine node_component_hothalocoldmode_read_raw
subroutine node_component_hothalonull_destroy
subroutine node_component_hothalonull_initializor
subroutine node_component_hothalonull_read_raw
subroutine node_component_hothalostandard_read_raw
subroutine node_component_hothaloverysimple_read_raw
subroutine node_component_indicesnull_read_raw
subroutine node_component_indicesstandard_read_raw
subroutine node_component_massflowstatisticsnull_read_raw
subroutine node_component_massflowstatisticsstandard_read_raw
subroutine node_component_mergingstatisticsnull_read_raw
subroutine node_component_mergingstatisticsrecent_read_raw

subroutine node_component_hothalooutflowtracking_destroy
subroutine node_component_hothalostandard_destroy
subroutine node_component_hothaloverysimple_destroy
subroutine node_component_indicesnull_destroy
subroutine node_component_indicesstandard_destroy
subroutine node_component_massflowstatisticsnull_destroy
subroutine node_component_massflowstatisticsstandard_destroy
subroutine node_component_mergingstatisticsnull_destroy
subroutine node_component_mergingstatisticsrecent_destroy

subroutine node_component_hothalooutflowtracking_initializer
subroutine node_component_hothalostandard_initializer
subroutine node_component_hothaloverysimple_initializer
subroutine node_component_indicesnull_initializer
subroutine node_component_indicesstandard_initializer
subroutine node_component_massflowstatisticsnull_initializer
subroutine node_component_massflowstatisticsstandard_initializer
subroutine node_component_mergingstatisticsnull_initializer
subroutine node_component_mergingstatisticsrecent_initializer

19.1. Program units
19. Source Code Documentation

subroutine node_component_-mergingstatisticsrecent_read_raw
subroutine node_component_-mergingstatisticsstandard_destroy
subroutine node_component_-mergingstatisticsstandard_read_raw
subroutine node_component_nbodynull_-read_raw
subroutine node_component_nbodynull_destroy
subroutine node_component_nbodynull_initializor
subroutine node_component_nbodystandard_destroy
subroutine node_component_nbodystandard_initializor
subroutine node_component_nbodystandard_read_raw
subroutine node_component_positionnull_destroy
subroutine node_component_positionnull_initializor
subroutine node_component_positionnull_read_raw
subroutine node_component_positionpreset_destroy
subroutine node_component_positionpreset_initializor
subroutine node_component_positionpreset_read_raw
subroutine node_component_satellitenull_destroy
subroutine node_component_satellitenull_initializor
subroutine node_component_satellitenull_read_raw
subroutine node_component_satelliteorbiting_destroy
subroutine node_component_satelliteorbiting_initializor
subroutine node_component_satelliteorbiting_read_raw
subroutine node_component_satellitepreset_destroy
subroutine node_component_satellitepreset_initializor
subroutine node_component_satellitepreset_read_raw
subroutine node_component_satellitestandard_destroy
subroutine node_component_satellitestandard_initializor
subroutine node_component_satellitestandard_read_raw
subroutine node_component_satelliteverysimple_destroy
subroutine node_component_satelliteverysimple_initializor
subroutine node_component_satelliteverysimple_read_raw
subroutine node_component_spheroidnull_destroy
subroutine node_component_spheroidnull_initializor
subroutine node_component_- spheroidnull_read_raw 
subroutine node_component_- spheroidstandard_destroy 
subroutine node_component_- spheroidstandard_read_raw 
subroutine node_component_spinnull_- destroy 
subroutine node_component_spinnull_- read_raw 
subroutine node_component_- spinpreset3d_destroy 
subroutine node_component_- spinpreset3d_read_raw 
subroutine node_component_spinpreset_- destroy 
subroutine node_component_spinpreset_- read_raw 
subroutine positionpresetpositionset 
subroutine positionpresetvelocityset 
subroutine satelliteorbitingboundmassset 
subroutine satelliteorbitingmergetimeset 
subroutine satelliteorbitingpositionset 
subroutine satelliteorbitingtidaltensorpathintegratedset 
subroutine satelliteorbitingvirialorbitset 
subroutine satellitepresetnodeindexhistoryset 
subroutine satellitepresetvirialorbitset 
subroutine satellitestandardmergetimeset 
subroutine satelliteverysimplemergetimeset 
subroutine serializetoarrayrates 
subroutine serializetoarrayscales 
subroutine spheroidstandardabundancesgasset 
subroutine spheroidstandardabundancesstellarset 
subroutine spheroidstandardangularmomentumset
subroutine spheroidstandardluminositiesstellarset subroutine spheroidstandardmassstellarset subroutine spheroidstandardstarformationhistoryset subroutine spheroidstandardvelocityset subroutine spinpreset3dspinvectorset subroutine spinpresetspinset subroutine node_component_black_hole_standard_output_properties subroutine node_component_merging_statistics_recent_initialize subroutine node_component_spheroid_standard_star_formation_history_rate subroutine stellar_luminosities_create subroutine stellar_luminosities_destructor subroutine stellar_luminosities_initialize subroutine table_1d_destroy subroutine table_linear_1d_create subroutine table_linear_cspline_1d_destroy subroutine radiation_set_cmb subroutine radiation_set_time_cmb subroutine radiation_igb_file_initialize subroutine radiation_igb_internal_set function imf_energy_input_rate_noninstantaneous function imf_recycling_rate_noninstantaneous subroutine star_formation_imf_initialize subroutine stellar_astrophysics_file_initialize subroutine stellar_tracks_initialize_file subroutine stellar_population_properties_scales_noninstantaneous subroutine spheroidstandardmassgasset subroutine spheroidstandardradiusset subroutine spheroidstandardstellarpropertieshistoryset subroutine spinpreset3dspinvectorgrowthrateset subroutine spinpresetspingrowthrateset subroutine spinrandomspinset subroutine node_component_disk_exponential_initialize subroutine node_component_spheroid_standard_initialize subroutine radiation_define subroutine stellar_luminosities_destroy subroutine stellar_luminosities_expand_filter_set subroutine stellar_luminosities_special_cases subroutine table_generic_1d_create subroutine table_linear_cspline_1d_create subroutine tensor_r2_d3_sym_destroy subroutine power_spectrum_compute subroutine radiation_set_time_cmb subroutine radiation_igb_file_set subroutine radiation_intergalactic_background_internal_initialize subroutine star_formation_imf_initialize subroutine mass_function_covariance_matrix subroutine supernovae_type_ia_nagashima_initialize subroutine stellar_population_luminosity subroutine stellar_population_spectra_conroy_initialize_imf
19.1. Program units

subroutine stellar_population_spectra_file_read
subroutine excursion_sets_barrier_initialize
function excursion_sets_first_crossing_probability_farahi
function excursion_sets_first_crossing_probability_zhang_hui
subroutine linear_growth_state_retrieve
function power_spectrum_nonlinear_cosmicemu
subroutine power_spectrum_power_law_tabulate
subroutine transfer_function_eisenstein_hu_make
subroutine transfer_function_file_read
subroutine linear_growth_factor_simple_tabulate
program tests_io_hdf5
program test_nfw96_concentration_dark_energy
program test_zhao2009_flat
program test_zhao2009_open
program tests_bug745815
program tests_cosmic_age
program test_hashes
program test_perfect_hashes
program tests_linear_growth_ed
program tests_linear_growth_dark_energy
program test_nodes
program test_random
program tests_spherical-collapse_dark_energy_ed
program tests_spherical-collapse_dark_energy_omega_zero_point_six
program tests_spherical-collapse_dark_energy_omega_zero_point_eight
program tests_spherical-collapse_dark_energy_omega_two_thirds
program tests_spherical-collapse_dark_energy_open
program tests_spherical-collapse_open
program tests_tree_branch_destroy
subroutine stellar_population_spectra_file_tabulation
subroutine excursion_sets_first_crossing_farahi_read_file
subroutine excursion_sets_first_crossing_rate_tabulate_farahi
function excursion_sets_first_crossing_probability_zhang_hui_high
subroutine linear_growth_factor_simple_tabulate
subroutine primordial_power_spectrum_state_retrieve
subroutine transfer_function_bbks_make
subroutine transfer_function_state_retrieve
subroutine transfer_function_null_make
program tests_io_xml
program test_prada2011_concentration
program test_zhao2009_dark_energy
program test_abundances
program tests_comoving_distance
program tests_halo_mass_function_tinker
program test_hashes_cryptographic
program tests_kepler_orbits
program tests_linear_growth_cosmological_constant
program test_mass_distributions
program tests_power_spectrum
program tests_sigma
program tests_spherical-collapse_dark_energy_omega_zero_point_six
program tests_spherical-collapse_dark_energy_omega_half
program tests_spherical-collapse_dark_energy_lambda
program tests_spherical-collapse_flat
program test_tables
subroutine io_hdf5_read_attribute_character_1d_array_allocatable
subroutine io_hdf5_read_attribute_double_1d_array_allocatable
subroutine io_hdf5_read_attribute_integer8_1d_array_allocatable
subroutine io_hdf5_read_attribute_integer8_1d_array_static
subroutine io_hdf5_read_attribute_integer_1d_array_allocatable
subroutine io_hdf5_read_attribute_varstring_1d_array_allocatable_do_read
subroutine io_hdf5_read_dataset_character_1d_array_allocatable
subroutine io_hdf5_read_dataset_double_1d_array_allocatable
subroutine io_hdf5_read_dataset_double_2d_array_allocatable
subroutine io_hdf5_read_dataset_double_3d_array_allocatable
subroutine io_hdf5_read_dataset_double_4d_array_allocatable
subroutine io_hdf5_read_dataset_double_5d_array_allocatable
subroutine io_hdf5_read_dataset_integer8_1d_array_allocatable
subroutine io_hdf5_read_dataset_integer8_1d_array_static
subroutine io_hdf5_read_dataset_integer_1d_array_allocatable
subroutine io_hdf5_read_dataset_varstring_1d_array_allocatable_do_read
subroutine io_hdf5_write_attribute_integer8_1d
subroutine io_hdf5_write_dataset_integer8_1d
subroutine xml_array_read_one_column
subroutine xml_array_read_two_column
subroutine xml_list_array_read_one_column
subroutine hash_perfect_create
subroutine hash_perfect_destroy
function get_new_assert_result
subroutine unit_tests_finish

subroutine: add_memory_component
Description: Add a memory type to the memory reporting strings.
Code lines: 22
Contained by: module memory_management
Modules used: iso_varying_string

interface: alloc_array
Description: Generic interface to routines which allocate arrays.
Code lines: 26
Contained by: module memory_management

subroutine: alloc_array_character_1d
Description: Allocate a 1D character array.
Code lines: 40
Contained by: module memory_management
Modules used: dmemory
iso_varying_string
galacticus_display
string_handling

subroutine: alloc_array_character_1d_kind_int8
Description: Allocate a 1D character array.
Code lines: 40
Contained by: module memory_management
Modules used: dmemory
galacticus_display
19.1. Program units

```plaintext

**iso_varying_string**  **string_handling**

**subroutine:** alloc_array_double_precision_1d
**Description:** Allocate a 1D double_precision array.
**Code lines:** 40
**Contained by:** module memory_management
**Modules used:**
  - dmemory
  - iso_varying_string

**subroutine:** alloc_array_double_precision_1d_kind_int8
**Description:** Allocate a 1D double_precision array.
**Code lines:** 40
**Contained by:** module memory_management
**Modules used:**
  - dmemory
  - iso_varying_string

**subroutine:** alloc_array_double_precision_2d
**Description:** Allocate a 2D double_precision array.
**Code lines:** 40
**Contained by:** module memory_management
**Modules used:**
  - dmemory
  - iso_varying_string

**subroutine:** alloc_array_double_precision_2d_kind_int8
**Description:** Allocate a 2D double_precision array.
**Code lines:** 40
**Contained by:** module memory_management
**Modules used:**
  - dmemory
  - iso_varying_string

**subroutine:** alloc_array_double_precision_3d
**Description:** Allocate a 3D double_precision array.
**Code lines:** 40
**Contained by:** module memory_management
**Modules used:**
  - dmemory
  - iso_varying_string

**subroutine:** alloc_array_double_precision_3d_kind_int8
**Description:** Allocate a 3D double_precision array.
**Code lines:** 40
**Contained by:** module memory_management
**Modules used:**
  - dmemory
  - iso_varying_string

**subroutine:** alloc_array_double_precision_4d
**Description:** Allocate a 4D double_precision array.
**Code lines:** 40
**Contained by:** module memory_management
```
19. Source Code Documentation

**Modules used:**
- dmemory
- iso_varying_string
- galacticus_display
- string_handling

**subroutine:** alloc_array_double_precision_4d_kind_int8

*Description:* Allocate a 4D double_precision array.
*Code lines:* 40
*Contained by:* module memory_management
*Modules used:*
- dmemory
- iso_varying_string
- galacticus_display
- string_handling

**subroutine:** alloc_array_double_precision_5d

*Description:* Allocate a 5D double_precision array.
*Code lines:* 40
*Contained by:* module memory_management
*Modules used:*
- dmemory
- iso_varying_string
- galacticus_display
- string_handling

**subroutine:** alloc_array_double_precision_5d_kind_int8

*Description:* Allocate a 5D double_precision array.
*Code lines:* 40
*Contained by:* module memory_management
*Modules used:*
- dmemory
- iso_varying_string
- galacticus_display
- string_handling

**subroutine:** alloc_array_integer_1d

*Description:* Allocate a 1D integer array.
*Code lines:* 40
*Contained by:* module memory_management
*Modules used:*
- dmemory
- iso_varying_string
- galacticus_display
- string_handling

**subroutine:** alloc_array_integer_1d_kind_int8

*Description:* Allocate a 1D integer array.
*Code lines:* 40
*Contained by:* module memory_management
*Modules used:*
- dmemory
- iso_varying_string
- galacticus_display
- string_handling

**subroutine:** alloc_array_integer_2d

*Description:* Allocate a 2D integer array.
*Code lines:* 40
*Contained by:* module memory_management
*Modules used:*
- dmemory
- iso_varying_string
- galacticus_display
- string_handling

**subroutine:** alloc_array_integer_2d_kind_int8

*Description:* Allocate a 2D integer array.
*Code lines:* 40
*Contained by:* module memory_management
19.1. Program units

**subroutine:** alloc_array_integer_kind_int8_1d
*Description:* Allocate a 1D integer array.
*Code lines:* 40
*Contained by:* module memory_management
*Modules used:*

subroutine: alloc_array_integer_kind_int8_1d_kind_int8
*Description:* Allocate a 1D integer array.
*Code lines:* 40
*Contained by:* module memory_management
*Modules used:*

subroutine: alloc_array_integer_kind_int8_2d
*Description:* Allocate a 2D integer array.
*Code lines:* 40
*Contained by:* module memory_management
*Modules used:*

subroutine: alloc_array_integer_kind_int8_2d_kind_int8
*Description:* Allocate a 2D integer array.
*Code lines:* 40
*Contained by:* module memory_management
*Modules used:*

subroutine: alloc_array_logical_1d
*Description:* Allocate a 1D logical array.
*Code lines:* 40
*Contained by:* module memory_management
*Modules used:*

subroutine: alloc_array_logical_1d_kind_int8
*Description:* Allocate a 1D logical array.
*Code lines:* 40
*Contained by:* module memory_management
*Modules used:*

subroutine: alloc_array_real_1d
*Description:* Allocate a 1D real array.
*Code lines:* 40
*Contained by:* module memory_management
*Modules used:*

**Modules used:**

dmemory
iso_varying_string
galacticus_display
string_handling
19. Source Code Documentation

Modules used: dmemory
galacticus_display
iso_varying_string
string_handling

subroutine: alloc_array_real_1d_kind_int8
Description: Allocate a 1D real array.
Code lines: 40
Contained by: module memory_management
Modules used: dmemory
galacticus_display
iso_varying_string
string_handling

subroutine: code_memory_usage
Description: If present reads the file \((executable).size\) to determine the amount of memory the \((executable).exe\) code needs before other memory is allocated. This is stored to allow an accurate calculation of the memory used by the code. The \((executable).size\) file is made by running the Perl script Find_Executable_Size.pl (which is done automatically when the executable is built by make).
Code lines: 37
Contained by: module memory_management
Modules used: dmemory
galacticus_display
iso_varying_string

interface: dealloc_array
Description: Generic interface to routines which deallocate arrays.
Code lines: 14
Contained by: module memory_management

subroutine: dealloc_array_character_1d
Description: Deallocate a 1D character array.
Code lines: 31
Contained by: module memory_management
Modules used: dmemory
galacticus_display
iso_varying_string
string_handling

subroutine: dealloc_array_double_precision_1d
Description: Deallocate a 1D double precision array.
Code lines: 31
Contained by: module memory_management
Modules used: dmemory
galacticus_display
iso_varying_string
string_handling

subroutine: dealloc_array_double_precision_2d
Description: Deallocate a 2D double precision array.
Code lines: 31
Contained by: module memory_management
Modules used: dmemory
galacticus_display
iso_varying_string
string_handling

subroutine: dealloc_array_double_precision_3d
Description: Deallocate a 3D double precision array.
19.1. Program units

subroutine: dealloc_array_double_precision_4d
Description: Deallocate a 4D double_precision array.
Code lines: 31
Contained by: module memory_management
Modules used: dmemory
iso_varying_string
galacticus_display
string_handling

subroutine: dealloc_array_double_precision_5d
Description: Deallocate a 5D double_precision array.
Code lines: 31
Contained by: module memory_management
Modules used: dmemory
iso_varying_string
galacticus_display
string_handling

subroutine: dealloc_array_integer_1d
Description: Deallocate a 1D integer array.
Code lines: 31
Contained by: module memory_management
Modules used: dmemory
iso_varying_string
galacticus_display
string_handling

subroutine: dealloc_array_integer_2d
Description: Deallocate a 2D integer array.
Code lines: 31
Contained by: module memory_management
Modules used: dmemory
iso_varying_string
galacticus_display
string_handling

subroutine: dealloc_array_integer_kind_int8_1d
Description: Deallocate a 1D integer array.
Code lines: 31
Contained by: module memory_management
Modules used: dmemory
iso_varying_string
galacticus_display
string_handling

subroutine: dealloc_array_integer_kind_int8_2d
Description: Deallocate a 2D integer array.
Code lines: 31
Contained by: module memory_management
Modules used: dmemory
iso_varying_string
galacticus_display
string_handling

subroutine: dealloc_array_logical_1d
19. Source Code Documentation

**Description:** Deallocate a 1D logical array.
**Code lines:** 31
**Contained by:** module memory_management
**Modules used:**
- dmemory
- iso_varying_string
- galacticus_display
- string_handling

**subroutine:** dealloc_array_real_1d

**Description:** Deallocate a 1D real array.
**Code lines:** 31
**Contained by:** module memory_management
**Modules used:**
- dmemory
- iso_varying_string
- galacticus_display
- string_handling

**function:** memory_usage_get

**Code lines:** 9
**Contained by:** module memory_management
**Modules used:**
- dmemory

**subroutine:** memory_usage_record

**Description:** Record a change in memory usage.
**Code lines:** 41
**Contained by:** module memory_management
**Modules used:**
- dmemory
- iso_c_binding
- iso_varying_string
- string_handling

**subroutine:** memory_usage_report

**Description:** Writes a report on the current memory usage. The total memory use is evaluated and all usages are scaled into convenient units prior to output.
**Code lines:** 54
**Contained by:** module memory_management
**Modules used:**
- dmemory
- iso_varying_string
- galacticus_display

**type:** memoryusage

**Description:** Derived type variable for storing the properties of a single class of memory storage (memory usage, divisor for outputting and suffix for outputting)
**Code lines:** 6
**Contained by:** module memory_management

**type:** memoryusagelist

**Description:** Derived type variable for storing all memory usage in the code.
**Code lines:** 3
**Contained by:** module memory_management

**subroutine:** set_memory_prefix

**Description:** Given a memory variable, sets the divisor and suffix required to put the memory usage into convenient units for output.
**Code lines:** 29
**Contained by:** module memory_management
file: utility.memory_usage.cpp

file: utility.regular_expressions.F90
   Description: Contains a module which implements regular expressions by wrapping the GNU C Library implementations.
   Code lines: 131

module: regular_expressions
   Description: Implements regular expressions by wrapping the GNU C Library implementations.
   Code lines: 108
   Contained by: file utility.regular_expressions.F90
   Modules used: iso_c_binding
   Used by: subroutine sussingopen subroutine sussingopen
            program tests_regular_expressions subroutine input_parameters_file_open

interface: regex
   Description: Constructor for regular expression object.
   Code lines: 3
   Contained by: module regular_expressions

function: regular_expression_constructor
   Description: Constructor for regEx objects.
   Code lines: 8
   Contained by: module regular_expressions

subroutine: regular_expression_destroy
   Description: Destroy a regEx object.
   Code lines: 10
   Contained by: module regular_expressions

subroutine: regular_expression_destructor
   Description: Destructor for regEx objects.
   Code lines: 8
   Contained by: module regular_expressions

function: regular_expression_match
   Description: Returns true if a regEx object matches the supplied string.
   Code lines: 10
   Contained by: module regular_expressions

file: utility.semaphores.F90
   Description: Contains a module which implements semaphores.
   Code lines: 198

module: semaphores
   Description: Implements semaphores.
   Code lines: 175
   Contained by: file utility.semaphores.F90
19. Source Code Documentation

Modules used: iso_c_binding
Used by: module galacticus_error function galacticus_task_evolve_tree

**type:** semaphore
**Code lines:** 36
**Contained by:** module semaphores

**subroutine:** semaphore_close
**Code lines:** 6
**Contained by:** module semaphores

**function:** semaphore_open
**Code lines:** 26
**Contained by:** module semaphores

**subroutine:** semaphore_post
**Code lines:** 8
**Contained by:** module semaphores

**subroutine:** semaphore_post_on_error
**Description:** Attempts to post to all open semaphores before exiting the code in an error condition.
**Code lines:** 13
**Contained by:** module semaphores

**subroutine:** semaphore_unlink
**Code lines:** 6
**Contained by:** module semaphores

**subroutine:** semaphore_wait
**Code lines:** 8
**Contained by:** module semaphores

**type:** semaphorelist
**Code lines:** 3
**Contained by:** module semaphores

**file:** utility.stateful_values.F90
**Description:** Contains a module of stateful types.
**Code lines:** 40

**module:** stateful_types
**Description:** Contains stateful types.
**Code lines:** 20
**Contained by:** file utility.stateful_values.F90
**Used by:** module merger_tree_read_importers file merger_-trees.construct.read.importer.SussingMergerTrees

**file:** merger_-trees.construct.read.importer.galacticus.F90
19.1. Program units

type: statefuldouble
  Code lines: 3
  Contained by: module stateful_types

type: statefulinteger
  Code lines: 3
  Contained by: module stateful_types

type: statefullogical
  Code lines: 3
  Contained by: module stateful_types

file: utility.string_handling.F90
  Description: Contains a module which implements various useful functionality for manipulating character strings.
  Code lines: 396

module: string_handling
  Description: Implements various useful functionality for manipulating character strings.
  Code lines: 376
  Contained by: file utility.string_handling.F90
  Modules used: iso_varying_string
  Used by: function hopkins2007constructor
  function abundance_pattern_lookup
  subroutine atomic_data_initialize
  subroutine cooling_function_not_matched
  subroutine cooling_radius_simple_initialize
  subroutine events_node_merger_do_slh
  function galactic_structure_radius_enclosing_mass
  function galacticus_component_list
  subroutine galacticus_output_analysis_mass_dpndnt_sz_dstrbtps
  subroutine galacticus_output_analysis_mass_functions
  function galacticus_build_string
  subroutine make_output_group
  subroutine galacticus_output_tree_rotation_curve_initialize
  subroutine galacticus_output_tree_velocity_dispersion_initialize
  function hopkins2007constructor
  function atom_lookup
  subroutine chemical_state_atomic_cie_cloudy_create
  subroutine cooling_function_atomic_cie_cloudy_create
  function node_branch_jump
  function node_subhalo_promotion
  subroutine solve_for_radius
  subroutine galacticus_output_tree_rotation_curve_initialize
  subroutine galacticus_output_tree_velocity_dispersion_initialize
  subroutine galacticus_output_analysis_mass_dpndnt_sz_dstrbtps
  subroutine galacticus_output_analysis_mass_functions
  function galacticus_build_string
  subroutine make_output_group
  subroutine galacticus_output_tree_rotation_curve_initialize
  subroutine galacticus_output_tree_velocity_dispersion_initialize
  function galacticus_version

1977
subroutine galacticus_state_retrieve
function galacticus_task_evolve_tree
subroutine geometry_survey_window_functions_caputi_2011_ukidss_uds
subroutine geometry_survey_window_functions_li_white_2009_sdss
subroutine filter_response_load
subroutine assign_mergers
subroutine build_isolated_parent_pointers
subroutine build_subhalo_mass_histories
subroutine enforce_subhalo_status
subroutine scan_for_branch_jumps
subroutine time_until_merging_subresolution
subroutine sussingtreeindicesread
function evolve_to_time
subroutine events_node_merger
subroutine tree_node_merge
subroutine evolve_to_time_report
subroutine merger_tree_mass_accretion_history_output
subroutine odeiv2_solve
subroutine merger_tree_data_structure_export
subroutine merger_tree_data_structure_read_ascii
subroutine agestatisticscreatelinked
subroutine blackholecreatelinked
diskcreatelinked
formationtimecreatelinked
hothalocreatelinked
interoutputcreatelinked
mergingstatisticscreatelinked
node_component_agestatisticsnull_dump
node_component_agestatisticsstandard_dump
node_component_basicnoneyielding_dump
node_component_basicstandard_dump
node_component_basicstandardtracking_dump
node_component_basicstandardnull_dump
node_component_merger_tree_state_restore
node_component_merge_tree_evolve_to
node_component_history_error_handler
node_component_tree_node_odes_error_handler
node_component_merger_tree_record_evolution_output
node_component_merger_tree_read_do
node_component_sussingopen
node_component_sussingtreeindicesread
node_component_sussingtreeindicesread
node_component_merger_tree_structure_output
node_component_history_extend
node_component_merger_tree_data_structure_export
node_component_merger_tree_data_structure_read_ascii
node_component_basiccreate_linked
node_component_darkmatterprofilecreatelinked
dynamicsstatisticscreatelinked
hosthistorycreatelinked
indicescreatelinked
massflowstatisticscreatelinked
nbodycreatelinked
subroutine evolve_to_time_report
subroutine merger_tree_mass_accretion_history_output

1978
19.1. Program units

subroutine node_component_blackholenull_dump
subroutine node_component_blackholesimple_dump
subroutine node_component_blackholestandard_dump
subroutine node_component_darkmatterprofilenull_dump
subroutine node_component_darkmatterprofilescale_dump
subroutine node_component_darkmatterprofilescalepreset_dump
subroutine node_component_darkmatterprofilescaleshape_dump
subroutine node_component_diskexponential_dump
subroutine node_component_disknull_dump
subroutine node_component_diskverysimple_dump
subroutine node_component_dynamicsstatisticsbars_dump
subroutine node_component_dynamicsstatisticsnull_dump
subroutine node_component_formationtimecole2000_dump
subroutine node_component_hosthistorynull_dump
subroutine node_component_hosthistorystandard_dump
subroutine node_component_hothalo_coldmode_dump
subroutine node_component_hothalonull_dump
subroutine node_component_hothalooutflowtracking_dump
subroutine node_component_hothalostandard_dump
subroutine node_component_hothaloverysimple_dump
subroutine node_component_indicesnull_dump
subroutine node_component_indicesstandard_dump
subroutine node_component_interoutputnull_dump
subroutine node_component_interoutputstandard_dump
subroutine node_component_massflowstatisticsnull_dump
subroutine node_component_massflowstatisticsstandard_dump
subroutine node_component_mergingstatisticsnull_dump
subroutine node_component_mergingstatisticsrecent_dump
subroutine node_component_mergingstatisticsstandard_dump
subroutine node_component_nbodynull_dump
subroutine node_component_nbodystandard_dump
subroutine node_component_positionnull_dump
subroutine node_component_positionpreset_dump
subroutine node_component_satellitenull_dump
subroutine node_component_satelliteorbiting_dump
subroutine node_component_satellitepreset_dump
subroutine node_component_satellitestandard_dump
subroutine node_component_satelliteverysimple_dump
subroutine node_component_spheroidnull_dump
subroutine node_component_spheroidstandard_dump
subroutine node_component_spinpreset3d_dump
subroutine node_component_spinpreset_dump
subroutine node_component_spinrandom_dump
subroutine node_dump_xml
subroutine satellitecreatelinked
subroutine spincreatelinked
subroutine tree_node_remove_from_mergee

subroutine node_component_disk_exponential_post_evolve
subroutine node_component_dynamics_statistics_bars_output
subroutine stellar_luminosities_special_cases
subroutine satellite_merging_remnant_size_covington2008
subroutine stellar_tracks_initialize_file
subroutine stellar_population_spectra_conroy_initialize_imf
subroutine excursion_sets_barrier_initialize

program test_nfw96_concentration_dark_energy
program test_zhao2009_dark_energy
program test_string_utilities
subroutine io_hdf5_write_attribute_varstring_1d
function formatted_date_and_time
function get_input_parameter_array_size

subroutine get_input_parameter_double_c
subroutine get_input_parameter_varstring_array
subroutine alloc_array_character_1d
subroutine alloc_array_double_precision_1d
subroutine alloc_array_double_precision_2d
subroutine alloc_array_double_precision_3d
subroutine alloc_array_double_precision_4d

subroutine positioncreatelinked
subroutine spheroidcreatelinked
subroutine tree_node_remove_from_host
subroutine node_component_black_hole_standard_output_properties
subroutine node_component_disk_very_simple_post_evolve
subroutine node_component_spheroid_standard_post_evolve
subroutine satellite_merging_remnant_size_cole2000
subroutine satellite_move_to_new_host

function stellar_population_luminosity
function stellar_population_spectrum_postprocess_index
subroutine transfer_function_camb_make

program test_zhao2009_flat
program test_zhao2009_open
subroutine io_hdf5_close
subroutine io_hdf5_write_dataset_varstring_1d
function hash_md5
subroutine get_input_parameter_char_array
subroutine get_input_parameter_integer_c
subroutine input_parameters_file_open

subroutine alloc_array_character_1d_kind_int8
subroutine alloc_array_double_precision_1d_kind_int8
subroutine alloc_array_double_precision_2d_kind_int8
subroutine alloc_array_double_precision_3d_kind_int8
subroutine alloc_array_double_precision_4d_kind_int8
19.1. Program units

subroutine alloc_array_double_precision_5d
subroutine alloc_array_double_precision_5d_kind_int8
subroutine alloc_array_integer_1d
subroutine alloc_array_integer_1d_kind_int8
subroutine alloc_array_integer_2d
subroutine alloc_array_integer_2d_kind_int8
subroutine alloc_array_integer_kind_int8_1d
subroutine alloc_array_integer_kind_int8_1d_kind_int8
subroutine alloc_array_integer_kind_int8_2d
subroutine alloc_array_integer_kind_int8_2d_kind_int8
subroutine alloc_array_logical_1d
subroutine alloc_array_logical_1d_kind_int8
subroutine alloc_array_real_1d
subroutine alloc_array_real_1d_kind_int8
subroutine dealloc_array_character_1d
subroutine dealloc_array_double_precision_1d
subroutine dealloc_array_double_precision_2d
subroutine dealloc_array_double_precision_3d
subroutine dealloc_array_double_precision_4d
subroutine dealloc_array_double_precision_5d
subroutine dealloc_array_integer_1d
subroutine dealloc_array_integer_2d
subroutine dealloc_array_integer_kind_int8_1d
subroutine dealloc_array_integer_kind_int8_2d
subroutine dealloc_array_logical_1d
subroutine dealloc_array_real_1d
subroutine memory_usage_record
subroutine unit_tests_finish

function: concatenate_varstr_integer
Description: Provides a concatenation operator to append an integer number to a varying_string.
Code lines: 11
Contained by: module string_handling

function: concatenate_varstr_integer8
Description: Provides a concatenation operator to append an integer number to a varying_string.
Code lines: 12
Contained by: module string_handling
Modules used: kind_numbers

function: convert_varstring_to_char
Description: Convert an array of varying strings into an array of characters.
Code lines: 11
Contained by: module string_handling

interface: operator(/)
Code lines: 3
Contained by: module string_handling

function: string_c_to_fortran
Description: Convert a C-style character array into a Fortran varying string variable.
function: string_count_words
Description: Return a count of the number of space separated words in inputString.
Code lines: 28
Contained by: module string_handling

function: string_join
Description: Joins an array of strings into one long string with the given separator.
Code lines: 14
Contained by: module string_handling

function: string_levenshtein_distance
Description: Compute the Levenshtein distance between strings a and b.
Code lines: 27
Contained by: module string_handling

function: string_lower_case
Description: Converts an input string to lower case.
Code lines: 16
Contained by: module string_handling

interface: string_split_words
Code lines: 3
Contained by: module string_handling

subroutine: string_split_words_char
Description: Split inputString into words and return as an array.
Code lines: 46
Contained by: module string_handling

subroutine: string_split_words_varstring
Description: Split inputString into words and return as an array.
Code lines: 46
Contained by: module string_handling

function: string_strip
Description: Strips a string of leading and trailing whitespace, including tabs.
Code lines: 26
Contained by: module string_handling

function: string_subscript
Description: Converts an input string to Unicode subscripts.
Code lines: 16
Contained by: module string_handling

function: string_superscript
19.1. Program units

**Description:** Converts an input string to Unicode superscripts.
**Code lines:** 16
**Contained by:** module `string_handling`

**function: string_upper_case**

**Description:** Converts an input string to upper case.
**Code lines:** 16
**Contained by:** module `string_handling`

**function: string_upper_case_first**

**Description:** Converts an input string to upper case.
**Code lines:** 13
**Contained by:** module `string_handling`

**file: utility.unit_tests.F90**

**Description:** Contains a module which implements unit testing.
**Code lines:** 1014

**module: unit_tests**

**Description:** Implements unit testing.
**Code lines:** 994
**Contained by:** file `utility.unit_tests.F90`
**Modules used:** `iso_varying_string`
**Used by:**
- program `tests_io_hdf5`
- program `test_nfw96_concentration_dark_energy`
- program `test_prada2011_concentration`
- program `test_zhao2009_dark_energy`
- program `test_abundances`
- program `test_black_hole_fundamentals`
- program `tests_comoving_distance`
- program `tests_cosmic_age`
- program `tests_halo_mass_function_tinker`
- program `test_hashes`
- program `test_hashes_cryptographic`
- program `tests_integration`
- program `test_interpolation_2d`
- program `test_interpolation`
- program `tests_linear_growth_edm`
- program `tests_linear_growth_eds`
- program `tests_linear_growth_dark_energy`
- program `test_make_ranges`
- program `test_math_special_functions`
- program `test_nodes`
- program `tests_power_spectrum`
- program `tests_regular_expressions`
- program `test_search`
- program `tests_bug745815`
- program `test_coordinate_systems`
- program `test_kdekep_orbits`
- program `tests_linear_growth_cosmological_constant`
- program `tests_linear_growth_dark_energy`
- program `tests_linear_growth_open`
- program `tests_linear_growth_cosmological_constant`
- program `test_make_ranges`
- program `test_mass_distributions`
- program `test_meshes`
- subroutine `test_node_task`
- program `test_random`
- program `test_root_finding`
- program `tests_sigma`
19. Source Code Documentation

program test_sort
program tests_sphericalCollapse_
dark_energy_eds
program tests_sphericalCollapse_
dark_energy_omega_zero_point_eight
program tests_sphericalCollapse_
dark_energy_omega_two_thirds
program tests_sphericalCollapse_
dark_energy_open
program tests_sphericalCollapse_open
program test_tables
program tests_treeBranch_destroy
program test_vectors

program test_inoue2014
program tests_sphericalCollapse_
dark_energy_omega_zero_point_six
program tests_sphericalCollapse_
dark_energy_omega_half
program tests_sphericalCollapse_
dark_energy_omega_zero_point_eight
program tests_sphericalCollapse_
dark_energy_omega_two_thirds
program tests_sphericalCollapse_
dark_energy_lambda
program tests_sphericalCollapse_flat
program test_string_utilities
program test_tensors

interface: assert
Description: Generic interface for assert routines.
Code lines: 20
Contained by: module unit_tests

subroutine: assert_character_1d_array
Description: Assess and record an assertion about character arguments.
Code lines: 41
Contained by: module unit_tests

subroutine: assert_character_scalar
Description: Assess and record an assertion about character arguments.
Code lines: 41
Contained by: module unit_tests

subroutine: assert_double_1d_array
Description: Assess and record an assertion about double precision arguments.
Code lines: 49
Contained by: module unit_tests
Modules used: numerical_comparison

subroutine: assert_double_2d_array
Description: Assess and record an assertion about double precision arguments.
Code lines: 51
Contained by: module unit_tests
Modules used: numerical_comparison

subroutine: assert_double_3d_array
Description: Assess and record an assertion about double precision arguments.
Code lines: 53
Contained by: module unit_tests
Modules used: numerical_comparison

subroutine: assert_double_4d_array
Description: Assess and record an assertion about double precision arguments.
Code lines: 55
19.1. Program units

Contained by: module unit_tests
Modules used: numerical_comparison

**subroutine: assert_double_5d_array**
- **Description:** Assess and record an assertion about double precision arguments.
- **Code lines:** 57
- **Contained by:** module unit_tests
- **Modules used:** numerical_comparison

**subroutine: assert_double_scalar**
- **Description:** Assess and record an assertion about double precision arguments.
- **Code lines:** 43
- **Contained by:** module unit_tests
- **Modules used:** numerical_comparison

**subroutine: assert_integer8_1d_array**
- **Description:** Assess and record an assertion about integer arguments.
- **Code lines:** 42
- **Contained by:** module unit_tests
- **Modules used:** kind_numbers

**subroutine: assert_integer8_scalar**
- **Description:** Assess and record an assertion about integer arguments.
- **Code lines:** 42
- **Contained by:** module unit_tests
- **Modules used:** kind_numbers

**subroutine: assert_integer_1d_array**
- **Description:** Assess and record an assertion about integer arguments.
- **Code lines:** 41
- **Contained by:** module unit_tests

**subroutine: assert_integer_scalar**
- **Description:** Assess and record an assertion about integer arguments.
- **Code lines:** 41
- **Contained by:** module unit_tests

**subroutine: assert_logical_1d_array**
- **Description:** Assess and record an assertion about logical arguments.
- **Code lines:** 33
- **Contained by:** module unit_tests

**subroutine: assert_logical_scalar**
- **Description:** Assess and record an assertion about logical arguments.
- **Code lines:** 36
- **Contained by:** module unit_tests
- **Modules used:** galacticus_error
subroutine: assert_real_1d_array
Description: Assess and record an assertion about real arguments.
Code lines: 49
Contained by: module unit_tests
Modules used: numerical_comparison

subroutine: assert_real_scalar
Description: Assess and record an assertion about real arguments.
Code lines: 43
Contained by: module unit_tests
Modules used: numerical_comparison

subroutine: assert_varstring_1d_array
Description: Assess and record an assertion about character arguments.
Code lines: 41
Contained by: module unit_tests

subroutine: assert_varstring_scalar
Description: Assess and record an assertion about character arguments.
Code lines: 41
Contained by: module unit_tests

type: assertresult
Description: A derived type for storing results of asserts.
Code lines: 5
Contained by: module unit_tests

function: get_new_assert_result
Description: Get a new assert result object.
Code lines: 22
Contained by: module unit_tests
Modules used: memory_management

subroutine: unit_tests_begin_group
Description: Marks that a unit test group has begun.
Code lines: 10
Contained by: module unit_tests

subroutine: unit_tests_end_group
Description: Marks that a unit test group has ended.
Code lines: 8
Contained by: module unit_tests

subroutine: unit_tests_finish
Description: Write out the results of unit testing.
Code lines: 57
Contained by: module unit_tests
Modules used: galacticus_display memory_management
19.1. Program units

string_handling
Part V.

Contributions and Acknowledgements
20. Contributions

Contributions to the GALACTICUS project have been made by the following people:

Andrea Kulier
- ./perl/Galacticus/ColumnDensity.pm

Andrew Benson
- ./perl/Galacticus/ColumnDensity.pm
- black_holes.binaries.initial_radius.Volonteri_2003.F90
- black_holes.binaries.initial_radius.tidal_radius.F90
- black_holes.binaries.recoil_velocity.Campanelli2007.F90
- black_holes.binaries.recoil_velocity.F90
- black_holes.binaries.recoil_velocity.null.F90
- black_holes.binaries.separation_growth_rate.F90
- black_holes.binaries.separation_growth_rate.null.F90
- black_holes.binaries.separation_growth_rate.standard.F90
- dark_matter_profiles.structure.concentration.WDM.F90
- galactic_structure.potential.F90
- galactic_structure.rotation_curve.gradient.F90
- galactic_structure.velocity_dispersions.F90
- objects.nodes.components.black_hole.standard.structure_tasks.F90
- objects.nodes.components.satellite.orbiting.F90
- objects.nodes.components.satellite.orbiting.bound_functions.Inc
- objects.nodes.components.satellite.preset.F90
- objects.tensors.F90
- satellites.dynamical_friction.acceleration.Chandrasekhar1943.F90
- satellites.dynamical_friction.acceleration.F90
- satellites.dynamical_friction.acceleration.null.F90
- satellites.tidal_heating.rate.F90
- satellites.tidal_heating.rate.Gnedin1999.F90
- satellites.tidal_heating.rate.null.F90
- satellites.tidal_stripping.rate.F90
- satellites.tidal_stripping.rate.Zentner2005.F90
- satellites.tidal_stripping.rate.null.F90
- star_formation.rate_surface_density.disks.extended_Schmidt.F90
20. Contributions

- structure_formation.excursion_sets.first_crossing_distribution.Farahi.F90
- structure_formation.excursion_sets.first_crossing_distribution.Zhang_Hui_high_order.F90
- structure_formation.transfer_function.Eisenstein_Hu.F90

Anthony Pullen
- dark_matter_profiles.structure.concentration.WDM.F90
- objects.nodes.components.satellite.orbiting.F90
- objects.nodes.components.satellite.orbiting.bound_functions.Inc
- objects.tensors.F90
- satellites.dynamical_friction.acceleration.Chandrasekhar1943.F90
- satellites.dynamical_friction.acceleration.F90
- satellites.dynamical_friction.acceleration.null.F90
- satellites.tidal_heating.rate.F90
- satellites.tidal_heating.rate.Gnedin1999.F90
- satellites.tidal_heating.rate.null.F90
- satellites.tidalstripping.rate.F90
- satellites.tidalstripping.rate.Zentner2005.F90
- satellites.tidalstripping.rate.null.F90
- structure_formation.transfer_function.Eisenstein_Hu.F90

Arya Farahi
- star_formation.rate_surface_density.disks.extended_Schmidt.F90
- structure_formation.excursion_sets.first_crossing_distribution.Farahi.F90
- structure_formation.excursion_sets.first_crossing_distribution.Zhang_Hui_high_order.F90

Daniel McAndrew
- atomic.ionization_potentials.F90
- atomic.ionization_potentials.Verner.F90
- atomic.radiation.gaunt_factors.Sutherland1998.F90
- atomic.rates.excitation.collisional.F90
- atomic.rates.excitation.collisional.ScholzWalters91.F90
- atomic.rates.recombination.dielectronic.Arnaud85.F90
- atomic.rates.recombination.dielectronic.F90
- atomic.rates.recombination.radiative.data.F90
- intergalactic_medium.state.internal.F90
- intergalactic_medium.state.internal.evolver.F90

Jianling Gan

1992
- objects.nodes.components.satellite.preset.F90

Luiz Felippe S. Rodrigues
- intergalactic_medium.state.RecFast.F90
- intergalactic_medium.state.file.F90

Markus Haider
- satellites.merging.timescale.infinite.F90

Martin White
- satellites.merging.timescale.Wetzel-White.F90

Stephanie Dörschner
- Bolshoi_Merger_Tree_File_Maker.F90
- merger_trees.file_maker.Bolshoi.F90
- objects.merger_tree_data.F90

Stéphane Mangeon
- black_holes.binaries.initial_radius.Volonteri_2003.F90
- black_holes.binaries.initial_radius.tidal_radius.F90
- black_holes.binaries.recoil_velocity.Campanelli2007.F90
- black_holes.binaries.recoil_velocity.F90
- black_holes.binaries.recoil_velocity.null.F90
- black_holes.binaries.separation_growth_rate.F90
- black_holes.binaries.separation_growth_rate.null.F90
- black_holes.binaries.separation_growth_rate.standard.F90
- galactic_structure.potential.F90
- galactic_structure.rotation_curve.gradient.F90
- galactic_structure.velocity_dispersions.F90
- objects.nodes.components.black_hole.standard.structure_tasks.F90
21. Acknowledgements

In addition to the tools and libraries required to compile and run GALACTICUS, development of GALACTICUS has benefitted from extensive use of the following: GNU OCTAVE, MAXIMA, CANTOR, KILE, EMACS and VALGRIND. We are grateful to the members of the GNU FORTRAN mailing list for invaluable discussions and fixes for compiler problems. We thank John Burkardt for making available the BIVAR algorithm for performing interpolation on data irregularly spaced on a 2D plane and Dima Verner for making available codes to compute various atomic data for astrophysics. Chris Power provided instructions for installing GALACTICUS under Mac OS X. The community of GALACTICUS users\footnote{In particular, Christoph Behrens, Stephanie Dörschner, Jianling Gan, Markus Haider, Harald Höller, Eve Kovacs, Ting-Wen Lan, Adrian Pope, Luiz Felippe Rodrigues, Sergio Sanes, Martin White and Liyan Xu.} have provided invaluable feedback and bug reports. Gian Luigi Granato and Laura Silva kindly provided modifications to their GRASIL code to allow it to read GALACTICUS outputs.
Part VI.

Appendices
A. Merger Tree File Format

**Galacticus** uses a standardized HDF5 file structure for merger tree data. This allows for portability. This format is defined below.
A. Merger Tree File Format

A.1. Basic File Format

Merger trees are stored in HDF5 files for portability and convenience. Additionally, the format is intended to be sufficiently flexible to allow it to describe merger trees obtained in a wide variety of ways, including Monte Carlo algorithms (e.g. extended Press-Schechter algorithms) and from N-body simulations.

A.1.1. Flexibility and Extensibility

All of the groups/datasets in the file except for the treeIndex and haloTrees groups are, in principle, optional. This does not mean that a file created without some of these optional groups/datasets will be usable by a given code. It is the responsibility of a given code to check that all data that it requires is present in the file. You are therefore encouraged to include as much information as possible when constructing merger tree files.

Additionally, the file format is intended to be extensible. It is permissible to add additional datasets, for example to describe some other properties of nodes in each tree. Additional datasets should follow the structure of currently defined datasets, i.e. they should be stored as a single dataset combining all trees with nodes listed in the same order as for other datasets. For additional datasets which might be of general use you are encouraged to contact us and recommend them for inclusion in the standard—this allows their name to be standardized.

A.1.2. A Note on Scalar Attributes

Many of the HDF5 attributes discussed in this document are indicated to be scalar (rank 0) attributes. It is allowable within the standard that these be pseudo-scalars (rank 1 arrays containing a single element). This allows such attributes to be created using the h5lt API for example.

A.1.3. Example File Structure

An example of the structure of such a file, called “example.hdf5” is shown below using the format of h5dump. Each of the groups is described in detail in the following sections.

HDF5 "example.hdf5" {
  GROUP "/" {
    GROUP "cosmology" {
    }
    GROUP "groupFinder" {
    }
    GROUP "haloTrees" {
    }
    GROUP "mergerTrees" {
    }
    GROUP "particles" {
    }
    GROUP "provenance" {
    }
    GROUP "simulation" {
    }
    GROUP "treeIndex" {
    }
    GROUP "units" {

2000
A.2. Cosmology Group

The cosmology group describes the cosmological model within which the merger trees contained in the file were constructed. An example of this group, showing standard attributes, is given below.

```
GROUP "cosmology" {
    ATTRIBUTE "HubbleParam" {
        DATATYPE H5T_IEEE_F64LE
        DATASPACE SCALAR
        DATA {
            (0): 0.73
        }
    }
    ATTRIBUTE "OmegaMatter" {
        DATATYPE H5T_IEEE_F64LE
        DATASPACE SCALAR
        DATA {
            (0): 0.25
        }
    }
    ATTRIBUTE "OmegaLambda" {
        DATATYPE H5T_IEEE_F64LE
        DATASPACE SCALAR
        DATA {
            (0): 0.75
        }
    }
    ATTRIBUTE "OmegaBaryon" {
        DATATYPE H5T_IEEE_F64LE
        DATASPACE SCALAR
        DATA {
            (0): 0.045
        }
    }
    ATTRIBUTE "powerSpectrumIndex" {
        DATATYPE H5T_IEEE_F64LE
        DATASPACE SCALAR
        DATA {
            (0): 1
        }
    }
    ATTRIBUTE "sigma_8" {
        DATATYPE H5T_IEEE_F64LE
        DATASPACE SCALAR
        DATA {
            (0): 0.9
        }
    }
}
```
A. Merger Tree File Format

A.2.1. Standard Attributes

The following are standard attributes in the cosmology group (others may be added as desired).

**HubbleParam** The Hubble parameter in units of 100 km/s/Mpc at $z = 0$, $h_0$;

**OmegaMatter** The density of matter (both dark and baryonic matter) in units of the critical density at $z = 0$, $\Omega_M$;

**OmegaLambda** The density of dark energy in units of the critical density at $z = 0$, $\Omega_{\Lambda}$;

**OmegaBaryon** The density of matter (both dark and baryonic matter) in units of the critical density at $z = 0$, $\Omega_b$;

**powerSpectrumIndex** The index of the primordial power spectrum of matter fluctuations, i.e. $n_s$ for power spectrum $P(k) \propto k^{n_s}$;

**sigma_8** The root-variance of mass fluctuations in real space top-hat spheres of radius $8h^{-1}$Mpc computed from the $z = 0$ linear theory power spectrum, $\sigma_8$;

**transferFunction** A descriptor of the transfer function used to compute the power spectrum.

A.3. Group Finder Group

This group, typically relevant only for merger trees derived from N-body simulations, describes the characteristics of the group finding algorithm that was used to find halos in the simulation. An example of this group, showing standard attributes, is given below.

GROUP "groupFinder" {
    COMMENT "Group finder parameters."
    ATTRIBUTE "code" {
        DATATYPE H5T_STRING {
            STRSIZE 7;
            STRPAD H5T_STR_SPACEPAD;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
        }
        DATASPACE SCALAR
        DATA {
            (0): "CAMB"
        }
    }
}
A.3.1. Standard Attributes

The following are standard attributes in the groupFinder group (others may be added as desired).

code The name of the group finding code used in the construction of these merger trees;

linkingLength For friends-of-friends group finding algorithms the dimensionless (i.e. in units of the mean interparticle spacing) linking length used;

minimumParticleNumber The minimum number of particles that a group was required to have in order to be included in a merger tree.

A.4. Simulation Group

This group, typically relevant only for merger trees derived from N-body simulations, describes the characteristics of the simulation from which the trees were derived. An example of this group, showing standard attributes, is given below.

GROUP "simulation" {
  COMMENT "Simulation parameters."
  ATTRIBUTE "ErrTolIntAccuracy" {
    DATATYPE H5T_IEEE_F64LE
    DATASPACE SCALAR
    DATA {
      (0): 0.02
    }
  }
  ATTRIBUTE "TypeOfTimestepCriterion" {
A. Merger Tree File Format

```c
DATATYPE H5T_STD_I32LE
DATASPACE SCALAR
DATA {
 (0): 0
}
}

ATTRIBUTE "boxSize" {
  DATATYPE H5T_IEEE_F64LE
  DATASPACE SCALAR
  DATA {
 (0): 500
  }
}

ATTRIBUTE "code" {
  DATATYPE H5T_STRING {
    STRSIZE 8;
    STRPAD H5T_STR_SPACEPAD;
    CSET H5T_CSET_ASCII;
    CTYPE H5T_C_S1;
  }
  DATASPACE SCALAR
  DATA {
 (0): "GADGET-2"
  }
}

ATTRIBUTE "initialConditions" {
  DATATYPE H5T_STRING {
    STRSIZE 5;
    STRPAD H5T_STR_SPACEPAD;
    CSET H5T_CSET_ASCII;
    CTYPE H5T_C_S1;
  }
  DATASPACE SCALAR
  DATA {
 (0): "glass"
  }
}

ATTRIBUTE "softeningKernel" {
  DATATYPE H5T_STRING {
    STRSIZE 6;
    STRPAD H5T_STR_SPACEPAD;
    CSET H5T_CSET_ASCII;
    CTYPE H5T_C_S1;
  }
  DATASPACE SCALAR
  DATA {
 (0): "spline"
  }
}
```
A.4. Simulation Group

ATTRIBUTE "softeningPlummerEquivalent" {
    DATATYPE H5T_IEEE_F64LE
    DATASPACE SCALAR
    DATA {
        (0): 0.005
    }
}

ATTRIBUTE "startRedshift" {
    DATATYPE H5T_IEEE_F64LE
    DATASPACE SCALAR
    DATA {
        (0): 127
    }
}

A.4.1. Standard Attributes

The following are standard attributes in the simulation group (others may be added as desired).

boxSize Relevant for cubic volumes typical of cosmological simulations, this attribute gives the length of the box in whatever unit system the file used (see §A.5);

code The name of the code used to run the simulation;

initialConditions A description of the initial conditions;

softeningKernel A description of the softening kernel used;

softeningPlummerEquivalent The equivalent Plummer softening length;

startRedshift The redshift at which the simulation was begun.

GADGET-specific Standard Attributes

The following are standard attributes in the simulation group specifically relevant to simulations run with the GADGET code. They typically reflect the values of parameters used by that code.

ErrTolIntAccuracy The integration accuracy used by GADGET;

TypeOfTimestepCriterion The type of timestepping criterion used by GADGET;

SofteningGas Specifies the (comoving) softening of the first particle group in GADGET;

SofteningHalo Specifies the (comoving) softening of the second particle group in GADGET;

SofteningDisk Specifies the (comoving) softening of the third particle group in GADGET;

SofteningBulge Specifies the (comoving) softening of the fourth particle group in GADGET;

SofteningStars Specifies the (comoving) softening of the fifth particle group in GADGET;

SofteningBndry Specifies the (comoving) softening of the sixth particle group in GADGET;

SofteningGasMaxPhys Specifies the maximum physical softening of the first particle group in GADGET;
A. Merger Tree File Format

**SofteningHaloMaxPhys** Specifies the maximum physical softening of the second particle group in GADGET;

**SofteningDiskMaxPhys** Specifies the maximum physical softening of the third particle group in GADGET;

**SofteningBulgeMaxPhys** Specifies the maximum physical softening of the fourth particle group in GADGET;

**SofteningStarsMaxPhys** Specifies the maximum physical softening of the fifth particle group in GADGET;

**SofteningBdryMaxPhys** Specifies the maximum physical softening of the sixth particle group in GADGET.

### A.5. Units Group

This group describes the unit system used throughout the file. Attributes should be included for length, mass and velocity units. In each case, three attributes are required to describe the units used (in the following *quantity* refers to length, mass, time or velocity):

- **quantityUnitsInSI** The units of this quantity expressed in the SI system;
- **quantityHubbleExponent** The exponent of the reduced Hubble constant, $h$, appearing in the units for this quantity;
- **quantityScaleFactorExponent** The exponent, $n$, of the expansion factor, $a$, required to convert this quantity into physical units. That is, multiplying this quantity by $a^n$ will give the quantity in physical units.

For example, if lengths in the file are expressed in units of comoving $h^{-1}$ Mpc, then we would have

- `lengthUnitsInSI = 3.08568e+22`
- `lengthHubbleExponent = -1`
- `lengthScaleFactorExponent = 1`

This allows a code reading the data from a merger tree file to automatically convert it into whatever unit/coordinate system it chooses.

An example of this group, showing standard attributes, is given below.

```plaintext
ATTRIBUTE "lengthHubbleExponent" {
   DATATYPE H5T_STD_I32LE
   DATASPACE SCALAR
   DATA {
      (0): -1
   }
}

ATTRIBUTE "lengthScaleFactorExponent" {
   DATATYPE H5T_STD_I32LE
   DATASPACE SCALAR
   DATA {
      (0): 1
   }
}
```
ATTRIBUTE "lengthUnitsInSI" {
    DATATYPE H5T_IEEE_F64LE
    DATASPACE SCALAR
    DATA {
        (0): 3.08568e+22
    }
}
ATTRIBUTE "massHubbleExponent" {
    DATATYPE H5T_STD_I32LE
    DATASPACE SCALAR
    DATA {
        (0): -1
    }
}
ATTRIBUTE "massScaleFactorExponent" {
    DATATYPE H5T_STD_I32LE
    DATASPACE SCALAR
    DATA {
        (0): 0
    }
}
ATTRIBUTE "massUnitsInSI" {
    DATATYPE H5T_IEEE_F64LE
    DATASPACE SCALAR
    DATA {
        (0): 1.98892e+40
    }
}
ATTRIBUTE "timeHubbleExponent" {
    DATATYPE H5T_STD_I32LE
    DATASPACE SCALAR
    DATA {
        (0): 0
    }
}
ATTRIBUTE "timeScaleFactorExponent" {
    DATATYPE H5T_STD_I32LE
    DATASPACE SCALAR
    DATA {
        (0): 0
    }
}
ATTRIBUTE "timeUnitsInSI" {
    DATATYPE H5T_IEEE_F64LE
    DATASPACE SCALAR
    DATA {
        (0): 3.1556926e+16
    }
}
A.6. Halo Trees Group

The **haloTrees** group contains the data describing the actual merger trees. Nodes from each tree must be stored contiguously. An example of this group is given below. In this example, `<nodeCount>` is the total number of nodes in all merger trees.

```
GROUP "haloTrees" {
    ATTRIBUTE "haloMassesIncludeSubhalos" {
        DATATYPE H5T_STD_I32LE
        DATASPACE SCALAR
        DATA {
            (0): 0
        }
    }
    ATTRIBUTE "haloAngularMomentaIncludeSubhalos" {
        DATATYPE H5T_STD_I32LE
        DATASPACE SCALAR
        DATA {
            (0): 0
        }
    }
    ATTRIBUTE "treesAreSelfContained" {
        DATATYPE H5T_STD_I32LE
        DATASPACE SCALAR
        DATA {
            (0): 1
        }
    }
}
```
ATTRIBUTE "treesHaveSubhalos" {
  DATATYPE H5T_STD_I32LE
  DATASPACE SCALAR
  DATA {
    (0): 1
  }
}

ATTRIBUTE "velocitiesIncludeHubbleFlow" {
  DATATYPE H5T_STD_I32LE
  DATASPACE SCALAR
  DATA {
    (0): 0
  }
}

ATTRIBUTE "positionsArePeriodic" {
  DATATYPE H5T_STD_I32LE
  DATASPACE SCALAR
  DATA {
    (0): 0
  }
}

DATASET "descendentIndex" {
  COMMENT "The index of each descendent node."
  DATATYPE H5T_STD_I64LE
  DATASPACE SIMPLE { ( <nodeCount> ) / ( <nodeCount> ) }
}

DATASET "expansionFactor" {
  COMMENT "The expansion factor of each node."
  DATATYPE H5T_IEEE_F64LE
  DATASPACE SIMPLE { ( <nodeCount> ) / ( <nodeCount> ) }
}

DATASET "halfMassRadius" {
  COMMENT "The half mass radius of each node."
  DATATYPE H5T_IEEE_F64LE
  DATASPACE SIMPLE { ( <nodeCount> ) / ( <nodeCount> ) }
}

DATASET "hostIndex" {
  COMMENT "The index of each host node."
  DATATYPE H5T_STD_I64LE
  DATASPACE SIMPLE { ( <nodeCount> ) / ( <nodeCount> ) }
}

DATASET "nodeIndex" {
  COMMENT "The index of each node."
  DATATYPE H5T_STD_I64LE
  DATASPACE SIMPLE { ( <nodeCount> ) / ( <nodeCount> ) }
}

DATASET "nodeMass" {
A. Merger Tree File Format

The following are standard attributes in the *haloTrees* group (others may be added as desired).
haloMassesIncludeSubhalos Indicates whether or not the masses of halos include the masses of any subhalos that they may contain. A value of 0 implies that halo masses do not include masses of subhalos, while a value of 1 indicates that they do;

haloAngularMomentaIncludeSubhalos Indicates whether or not the angular momenta of halos include the angular momenta contributed by any subhalos that they may contain. A value of 0 implies that halo masses do not include the contribution of subhalos, while a value of 1 indicates that they do. If this attribute is not present it is assumed to be equal to the haloMassesIncludeSubhalos attribute;

treesAreSelfContained Indicates whether or not trees are self-contained, in the sense that nodes never transfer from one tree to another. A value of 0 implies that nodes can move from one tree to another, while a value of 1 implies that they can not;

treesHaveSubhalos Indicates whether or not trees contain information on subhalos. A value of 0 implies that they do not, while a value of 1 implies that they do. This attribute is a convenience, as subhalo presence can be determined from the node data directly;

velocitiesIncludeHubbleFlow Indicates whether or not velocities include the Hubble flow. A value of 0 indicates that they do not, while a value of 1 indicates that they do. See §11.2 for important notes on velocity definitions in Galacticus.

positionsArePeriodic Indicates whether or not positions are periodic (as in a cosmological cube simulation). A value of 0 indicates that they are not, while a value of 1 indicates that they are periodic, with a period of boxSize.

A.6.2. Standard Datasets

The following are standard datasets in the haloTrees group.

angularMomentum The angular momentum of the halo. This can be either the magnitude of the angular momentum or a 3-D vector;

expansionFactor The expansion factor (normalized to unity at the present day) at which this node is identified (note that only one of the expansionFactor, redshift and time datasets is required, since they are simply related, but multiple can be present);

descendentIndex The nodeIndex of the descendent of this node in the merger tree, or −1 if there is no descendent;

halfMassRadius The radius containing half the mass of the node;

hostIndex The nodeIndex of the node which hosts this node. For nodes that are self-hosting (i.e. that are not subhalos inside another halo), the value of hostIndex should be set equal to the node’s own nodeIndex;

nodeIndex An ID number for the node, unique at least within each tree. If nodes are able to move from one tree to another, the ID must be unique within all trees. No other constraints are placed on nodeIndex (e.g. it does not have to be monotonically increasing within the file for example);

nodeMass The mass of the node;

position The three dimensional position of this node;
A. Merger Tree File Format

redshift The redshift at which this node is identified (note that only one of the expansionFactor, redshift and time datasets is required, since they are simply related, but multiple can be present);

scaleRadius The characteristic scale radius in the node (typically, but not necessarily, the NFW scale radius);

spin The spin parameter, $\lambda$, of the halo. This can be either the spin magnitude or a 3-D vector;

time The time at which this node is identified (note that only one of the expansionFactor, redshift and time datasets is required, since they are simply related, but multiple can be present);

velocity The three dimensional velocity of the node. See §11.2 for important notes on velocity definitions in GALACTICUS;

particleIndex If the particles group is included, this dataset should give the index of the first entry in the particle datasets that corresponds to the particle associated with this node;

particleCount If the particles group is included, this dataset should give the number of entries in particle datasets that correspond to the particle associated with this node;

A.7. Tree Index Group

The treeIndex group contains indexing information which describes which sections of the datasets in the haloTrees group belong to each tree. An example of this group is given below.

GROUP "treeIndex" {
  DATASET "firstNode" {
    COMMENT "Position of the first node in each tree in the haloTrees datasets."
    DATATYPE H5T_STD_I32LE
    DATASPACE SIMPLE { ( <treeCount> ) / ( <treeCount> ) }
  }
  DATASET "numberOfNodes" {
    COMMENT "Number of nodes in each tree."
    DATATYPE H5T_STD_I32LE
    DATASPACE SIMPLE { ( <treeCount> ) / ( <treeCount> ) }
  }
  DATASET "treeIndex" {
    COMMENT "Unique index of tree."
    DATATYPE H5T_STD_I64LE
    DATASPACE SIMPLE { ( <treeCount> ) / ( <treeCount> ) }
  }
  DATASET "treeWeight" {
    COMMENT "The number of such trees required per unit volume to create a representative sample."
    DATATYPE H5T_STD_F64LE
    DATASPACE SIMPLE { ( <treeCount> ) / ( <treeCount> ) }
  }
}
A.7.1. Standard Datasets

`firstNode` For each tree, gives the position¹ in the `haloTrees` datasets of the first node in the tree (note that dataset indexing begins at 0). This *does not* necessarily have to be the root node of the tree—nodes in a single tree can be stored in any order in the `haloTrees` datasets, providing that they are contiguous;

`numberOfNodes` For each tree, gives the number of nodes in the tree;

`treeIndex` A unique ID number for each tree;

`treeWeight` A weight factor specifying the number density of each tree required to construct a representative sample. If not present, it is acceptable to assume that the weight is $1/\text{boxSize}^3$ if that attribute is present in the `simulation` group.

A.8. Merger Trees Group

The `mergerTrees` group is optional and provides a convenience method for accessing the properties of individual trees. If present, it contains one group for each tree in the file, named `mergerTree<treeID>` where `<treeID>` is the ID of the tree. Each of these groups should contain a set of scalar references to the sections of the datasets in the `haloTrees` group to which this tree corresponds. For example, the `descendentIndex` reference for the tree with ID number 89 would be as follows:

GROUP "mergerTrees/mergerTree89" {
    DATASET "descendentIndex" {
        DATATYPE H5T_REFERENCE
        DATASPACE SCALAR
        DATA {
            DATASET /haloTrees/descendentIndex {(<indexBegin>)-(<indexEnd>)}
        }
    }
}

A.9. Particles Group

The `particles` group is optional and contains information on particle trajectories. It is intended for use with merger tree derived from N-body simulations for which it is often useful to track the location of, for example, the most bound particle associated with a subhalo even after that subhalo can no longer be tracked in the simulation. An example of this group is given below. In this example, `<particleCount>` is the total number of particles included in the group.

GROUP "particles" {
    DATASET "particleID" {
        COMMENT "The ID of each particle."
        DATATYPE H5T_STD_I64LE
        DATASPACE SIMPLE { ( <particleCount> ) / ( <particleCount> ) }
    }
    DATASET "redshift" {

¹That is, it gives the array index of the first node of the tree in the `haloTrees/nodeIndex` dataset for example. It does not give the `nodeIndex` of the first node in the tree.
A. Merger Tree File Format

COMMENT "The redshift of each particle."
DATATYPE H5T_IEEE_F64LE
DATASPACE SIMPLE { ( <particleCount> ) / ( <particleCount> ) }
}
DATASET "time" {
COMMENT "The time of each particle."
DATATYPE H5T_IEEE_F64LE
DATASPACE SIMPLE { ( <particleCount> ) / ( <particleCount> ) }
}
DATASET "expansionFactor" {
COMMENT "The expansion factor of each particle."
DATATYPE H5T_IEEE_F64LE
DATASPACE SIMPLE { ( <particleCount> ) / ( <particleCount> ) }
}
DATASET "position" {
COMMENT "The position of each node."
DATATYPE H5T_IEEE_F64LE
DATASPACE SIMPLE { ( <particleCount>, 3 ) / ( <particleCount>, 3 ) }
}
DATASET "velocity" {
COMMENT "The velocity of each node."
DATATYPE H5T_IEEE_F64LE
DATASPACE SIMPLE { ( <particleCount>, 3 ) / ( <particleCount>, 3 ) }
}

Each particle should be stored contiguously (i.e. entries with the same particleID should be consecutive) and it is frequently convenient (although not required) that entries for each particle be arranged in order of increasing cosmic time.

A.9.1. Standard Datasets

particleID An ID, unique within the entire simulation, for each particle;

redshift The redshift at which the particle is recorded (a single particle can appear in these datasets at multiple times);

time The time at which the particle is recorded (a single particle can appear in these datasets multiple times at different redshifts);

expansionFactor The expansion factor at which the particle is recorded (a single particle can appear in these datasets multiple times at different expansion factors);

position The spatial position of the particle;

velocity The velocity of the particle. See §11.2 for important notes on velocity definitions in GALACTICUS.

Note that only one of the expansionFactor, time and redshift datasets is required, as they are simply related, but multiple of them can be present.
A.10. Merger Tree Builder

**Galacticus** contains software which builds merger tree files in the format described in §A.1 from merger tree descriptions in other formats, such as ASCII output from an SQL database. The merger tree building engine can be found in `source/objects.merger_tree_data.F90`. Examples of how this engine is used can be found in `source/Millennium_Merger_Tree_File_Maker.F90` and `source/merger_trees.file_maker.Millennium.F90` which are designed to work with the `scripts/aux/Millennium_Trees_Grab.pl` script to convert data extracted from the Millennium Simulation database into a format that **Galacticus** can read, and in `source/Simple_Merger_Tree_File_Maker.F90` and `source/merger_trees.file_maker.simple.F90` which are designed to work with ASCII file representations of mergers trees that contain just the mass, redshift and descendant of each node.

The basic process for building a merger tree file is to inform the engine of the data file to read and where specific information is located within that file. The data can then be processed and, finally, output in the required format. Specific interfaces that can be used are described below. Many of these interfaces work on an object `mergerTrees` of `mergerTreeData` type. This object stores all information on the merger trees while they are being internally processed.

**Setting property locations:** Before reading data from a file it is necessary to inform the tree builder engine of which column in the file corresponds to which property. This is done with repeated calls to `setProperty`, one for each column to read, as follows:

```fortran
    call mergerTrees%setProperty(propertyType,columnIndex)
```

where `columnIndex` is the number of the column (counting from 1) which contains the property specified by `propertyType`. `propertyType` can take one of the following values:

- `propertyTypeTreeIndex` A unique ID number for the tree to which this node belongs;
- `propertyTypeNodeIndex` An ID (unique within the tree) for this node;
- `propertyTypeDescendentIndex` The ID of the node’s descendant node;
- `propertyTypeHostIndex` The ID of the larger halo in which this node is hosted (equal to the node’s own ID if the node is self-hosting);
- `propertyTypeRedshift` The redshift of the node;
- `propertyTypeNodeMass` The mass of the node;
- `propertyTypeParticleCount` The number of particles in the node;
- `propertyTypePositionX` The $x$-position of the node (if present, both $y$ and $z$ components must also be present);
- `propertyTypePositionY` The $y$-position of the node (if present, both $x$ and $z$ components must also be present);
- `propertyTypePositionZ` The $z$-position of the node (if present, both $x$ and $y$ components must also be present);
- `propertyTypeVelocityX` The $x$-velocity of the node (if present, both $y$ and $z$ components must also be present);
- `propertyTypeVelocityY` The $y$-velocity of the node (if present, both $x$ and $z$ components must also be present);

2015
A. Merger Tree File Format

**propertyTypeVelocityZ** The z-velocity of the node (if present, both x and y components must also be present);

**propertyTypeSpinX** The x component of the node’s spin parameter (if present, both y and z components must also be present; cannot be present if spin magnitude is given);

**propertyTypeSpinY** The y component of the node’s spin parameter (if present, both x and z components must also be present; cannot be present if spin magnitude is given);

**propertyTypeSpinZ** The z component of the node’s spin parameter (if present, both x and y components must also be present; cannot be present if spin magnitude is given);

**propertyTypeSpin** The magnitude of the node’s spin parameter (cannot be present if spin vector components are given);

**propertyTypeAngularMomentumX** The x-component of the node’s angular momentum (if present, both y and z components must also be present; cannot be present if angular momentum magnitude is given);

**propertyTypeAngularMomentumY** The y-component of the node’s angular momentum (if present, both x and z components must also be present; cannot be present if angular momentum magnitude is given);

**propertyTypeAngularMomentumZ** The z-component of the node’s angular momentum (if present, both x and y components must also be present; cannot be present if angular momentum magnitude is given);

**propertyTypeAngularMomentum** The magnitude of the node’s angular momentum (cannot be present if angular momentum vector components are given);

**propertyTypeHalfMassRadius** The half-mass radius of the node;

**propertyTypeMostBoundParticleIndex** The index of the most bound particle in this node.

Not all properties must be specified—any required properties that are not specified will result in an error. Likewise, some properties, if present, require that other properties also be present. For example, if any of the position properties is given then all three positions are required.

*Reading ASCII data:* Once property columns have been specified, data from an ASCII file with one node per line can be read as follows:

```plaintext
call mergerTrees%readASCII(nodesFile,lineNumberStart,lineNumberStop,separator=",")
```

where `nodesFile` is the name of the file to read. The optional `lineNumberStart` and `lineNumberEnd` arguments give the first and last lines of the file to read, while the optional `separator` argument specifies the character used to separate columns (white space is assumed by default).

*Setting particle property locations:* If particle information is to be stored in the file, the locations of particle properties within the input file must be specified with repeated calls to `setParticleProperty` as follows:

```plaintext
call mergerTrees%setParticleProperty(propertyType,columnIndex)
```

where `columnName` is the number of the column (counting from 1) which contains the property specified by `propertyType`. `propertyType` can take one of the following values:
propertyTypeParticleIndex A unique ID for the particle;
propertyTypeRedshift The redshift of the particle;
propertyTypeNodeMass The mass of the particle;
propertyTypeParticleCount The number of particles in the particle;
propertyTypePositionX The x-position of the particle (if present, both y and z components must also be present);
propertyTypePositionY The y-position of the particle (if present, both x and z components must also be present);
propertyTypePositionZ The z-position of the particle (if present, both x and y components must also be present);
propertyTypeVelocityX The x-velocity of the particle (if present, both y and z components must also be present);
propertyTypeVelocityY The y-velocity of the particle (if present, both x and z components must also be present);
propertyTypeVelocityZ The z-velocity of the particle (if present, both x and y components must also be present).

Reading ASCII particle data: Once property columns have been specified, particle data from an ASCII file with one particle per line can be read as follows:

call mergerTrees%readParticlesASCII(particlesFile,lineNumberStart,lineNumberStop,separator="","")

where particlesFile is the name of the file to read. The optional lineNumberStart and lineNumberEnd arguments give the first and last lines of the file to read, while the optional separator argument specifies the character used to separate columns (white space is assumed by default).

Setting particle mass: The particle mass, particleMass, can be specified using:

call mergerTrees%setParticleMass(particleMass)

Specifying tree self-containment: Whether or not trees are self-contained can be specified using:

call mergerTrees%setSelfContained([.true.|.false.])

Specifying Hubble flow inclusion: Whether or not velocities include the Hubble flow can be specified using:

call mergerTrees%setIncludesHubbleFlow([.true.|.false.])

Specifying subhalo mass inclusion: Whether or not halo masses include the masses of any subhalos can be specified using:

call mergerTrees%setIncludesSubhaloMasses([.true.|.false.])

Specifying reference creation: Whether or not HDF5 reference to individual merger trees within the haloTrees datasets should be made can be specified using:

call mergerTrees%makeReferences([.true.|.false.])

Specifying units: The units used in the files can be specified with repeated calls to setUnits as follows:
A. Merger Tree File Format

call mergerTrees%setUnits(unitsType,unitsInSI,hubbleExponent,scaleFactorExponent)

where unitsType is one of:
unitsMass Units of mass;
unitsLength Units of length;
unitsTime Units of time;
unitsVelocity Units of velocity;

unitsInSI gives the units in the SI system, hubbleExponent specifies the power to which $h$ appears in the units and scaleFactorExponent specifies the number of powers of the expansion factor by which the quantity should be multiplied to place it into physical units.

Adding metadata: Meta-data can be added to the file by making repeated calls to addMetadata as follows:

call mergerTrees%addMetadata(metaDataType,label,value)

where metaDataType is one of:
metaDataGeneric Add to the generic metaData group;
metaDataCosmology Add to the cosmology group;
metaDataSimulation Add to the simulation group;
metaDataGroupFinder Add to the groupFinder group;
metaDataTreeBuilder Add to the treeBuilder group;
metaDataProvenance Add to the provenance group.

label is a label for this metadatum and value is the value to store. Currently integer, double precision and character data types are supported for metadata.

Exporting the data: Once the data has been read, units and properties specified and any metadata added, the trees can be exported to an HDF5 file using:

call mergerTrees%export(outputFile,outputFormat,hdfChunkSize,hdfCompressionLevel)

where outputFile is the name of the file to which the trees should be exported, outputFormat specifies the format to use (see §A.10.1), and hdfChunkSize and hdfCompressionLevel respectively give the chunk size and compression level to use when writing the file.

A.10.1. File Formats

The merger tree file builder engine can currently export in one of two formats. These formats can be specified on the command line to both Millennium_Merger_Tree_File_Maker.F90, and Simple_Merger_Tree_File_Maker.F90:
galacticus merger trees are exported in GALACTICUS’s native format described in detail in §A.1;
irate merger trees are exported in the IRATE format.
A.10.2. Exporting Trees from GALACTICUS

By setting [mergerTreesWrite]=true, GALACTICUS will export each merger tree generated to the file specified by [mergerTreeExportFileName] using the format specified by [mergerTreeExportOutputFormat]. Currently, node indices (plus host indices, which are assumed identical to the node indices), descendent indices, masses and redshifts are exported. Positions and velocities are exported if available. If IRATE-format output is requested then “snapshot” numbers will be assigned to nodes based on the time at which they exist. This usually only makes sense if the nodes are defined on a time grid (i.e. if merger trees were extracted from an N-body simulation, or if trees were re-gridded onto such a time grid; see §13.31.2). Export happens after any merger tree pre-evolution tasks (see §17.4.3).
B. Plotting Support

B.1. Plotting with **GNUPLOT**

While **GALACTICUS** data can, of course, be plotted using whatever method you choose, two Perl modules are provided that we find useful for plotting **GALACTICUS** data. These are intended for use with **GNUPLOT** and with datasets stored as **PDL** variables. The first module, **GnuPlot::PrettyPlots** plots lines and points with two color style (typically a lighter interior color and a darker border) with support for errorbars and limits (show as arrows) on points. The second, **GnuPlot::LaTeX** provides a convenient way to process output from **GNUPLOT**’s **epslatex** terminal into PDF files (suitable for inclusion in documents), PNG images with transparent backgrounds or **OpenOffice ODG** files (suitable for inclusion into presentations).

A typical use of these packages would look as follows:

```perl
use lib "./perl";
use PDL;
use GnuPlot::LaTeX;
use GnuPlot::PrettyPlots;

$outputFile = "myImage";
open($gnuPlot,"|gnuplot");
print $gnuPlot "set terminal epslatex color colortext lw 2 solid 7\n";
print $gnuPlot "set output '.\"$outputFile.\".eps'\n";
print $gnuPlot "set xlabel '\"x-axis label\"'\n";
print $gnuPlot "set ylabel '\"y-axis label\"'\n";
print $gnuPlot "set lmargin screen 0.15\n";
print $gnuPlot "set rmargin screen 0.95\n";
print $gnuPlot "set bmargin screen 0.15\n";
print $gnuPlot "set tmargin screen 0.95\n";
print $gnuPlot "set key spacing 1.2\n";
print $gnuPlot "set key at screen 0.4,0.8\n";
print $gnuPlot "set xrange [0.0:6.0]\n";
print $gnuPlot "set yrange [0.0:1.0]\n";
print $gnuPlot "set pointsize 2.0\n";
&PrettyPlots::Prepare_Dataset($plot,
    $x1Data, $y1Data,
    title => "First dataset",
    style => line,
    linePattern => 0,
    weight => [7,3],
    color => $PrettyPlots::colorPairs('lightGoldenrod')
);
```

1If you create an **OpenOffice ODG** file it’s recommended that you covert it to a Metafile within **OpenOffice** before putting it into a presentation—this seems to prevent a bug which occasionally causes an element of the plot to be lost during saving...
The process begins by opening a pipe to GNUPlot and specifying the `epslatex` terminal along with `color` and `colortext` options, any line weight preferences and the output EPS file. This is followed by commands to set up the plot, including labels, ranges etc. Note that you must specify margins manually\(^2\). Following this are calls to `PrettyPlots::Prepare_Dataset` which prepares instructions for plotting of a single dataset. The first argument is a reference to a structure which will store the instructions, while the second and third arguments are PDLs containing the \(x\) and \(y\) data to be plotted. Following this are multiple options as follows:

- **title** Gives the title of the dataset for inclusion in the plot key;
- **style** Specifies how the dataset should be drawn: either `line`, `point`, `boxes`, or `filledCurve`;
- **linePattern** Specifies the line pattern (as defined for GNUPlot’s `lt` option) to use;
- **symbol** A two element list giving the symbol indices that should be used to plot the border and inner parts of each point respectively;
- **weight** A two element list giving the line weights to be used for border and inner parts of each point/line respectively;
- **color** A two element list giving the color of border and inner parts of each point/line respectively. Colors should be specified as `#RRGGBB` in hexadecimal. Several suitable color pairs and sequences of pairs are defined in the `GnuPlot::PrettyPlots` module;
- **pointSize** Specifies the size of the points to be used;
- **errorNNN** Gives a PDL containing sizes of errors to be plotted on points in the up, down, left and right directions. A zero value will cause the error bar to be omitted, while a negative value will cause an arrow to be drawn with a length equal to the absolute value of the specified value;
- **filledCurve** If the `filledCurve` style is used, this option specifies the type of filled curve (closed, \(x_1\), \(x_2\), etc.—see the GNUPlot help `filledcurve` text for complete options). The default is `closed`;
- **y2** If the `filledCurve` style is used along with the `filledCurve=closed` option, this option is used to specify a second PDL of \(y\)-axis values. The region between this curve and the usual \(y\)-axis curve will be filled.

\(^2\)The `GnuPlot::PrettyPlots` module works by generating multiple layers of plotting which are overlaid. Axes are only drawn for the first layer. If you do not specify margins manually, they will be computed automatically for each layer and so will not match up between all layers. This will result in data being plotted incorrectly.
Once all datasets have been prepared, the call to &PrettyPlots::Plot_Datasets will generate the EPS and LaTeX files necessary to make the plot. This resulting plot can be converted to PDF, PNG or ODG form by calling &LaTeX::GnuPlot2PDF, &LaTeX::GnuPlot2PNG or &LaTeX::GnuPlot2ODG respectively. The EPS file will be replaced with the appropriate file. The &LaTeX::GnuPlot2PNG routine accepts an optional backgroundColor argument in #RRGGBB format. If present, this color will be used to set the background color of the plot (otherwise white is assumed). Although the background is made transparent in the PNG, setting the background color is important as antialiasing will make use of this background. Note that both PNG and ODG options will switch black axes and labels to white. Finally, the &LaTeX::GnuPlot2PNG routine accepts an optional margin argument which specifies the size of the margin (in pixels) to be left around the plot when cropping.

The ODG option requires that both pdf2svg and svg2office be installed on your system (svg2office should be located in /usr/local/bin).

### B.2. Merger Tree Diagrams with DOT

The dot command, which is a part of GraphViz is useful for creating diagrams of merger trees. Galacticus provides a function to output the structure of any merger tree in GraphViz format. This function, Merger_Tree_Dump, is provided by the Merger_Trees_Dump module. Usage is as follows:

```c
call Merger_Tree_Dump(
  &index,
  &baseNode,
  &highlightNodes =highlightNodes, &
  &backgroundColor ='white', &
  &nodeColor ='black', &
  &highlightColor ='black', &
  &edgeColor ='#DDDDDD', &
  &nodeStyle ='solid', &
  &highlightStyle ='filled', &
  &edgeStyle ='solid', &
  &labelNodes =.false., &
  &scaleNodesByLogMass =.true., &
  &edgeLengthsToTimes =.true., &
  &path ='/my/path' &
)
```

Here `index` is the tree index (successive calls to Merger_Tree_Dump with the same index will result in a sequence of output files—see below), and `baseNode` is a pointer to the base node of the tree to be dumped. All other arguments are optional:

- **highlightNodes** A list of node IDs. All nodes listed will be highlighted in the diagram;
- **backgroundColor** The color for the background of the diagram;
- **nodeColor** The color used to draw nodes;
- **highlightColor** The color used for highlighted nodes;
- **edgeColor** The color of edges (lines joining nodes);
- **nodeStyle** The style to use when drawing nodes;
- **labelNodes** A list of node IDs. All nodes listed will be highlighted in the diagram;
- **scaleNodesByLogMass** Whether or not to scale nodes by log mass;
- **edgeLengthsToTimes** Whether or not to scale edge lengths by time;
- **path** Path to save the output file to.

---

3This is just a personal preference for plots displayed in presentations—other options could be added.
B. Plotting Support

highlightStyle The style to use when drawing highlighted nodes;

edgeStyle The style to use when drawing edges;

labelNodes Specifies whether or not nodes should be labelled (labels consist of the node ID followed by the redshift);

scaleNodesByLogMass If true, the size of nodes will be set to be proportional to the logarithm of the node mass;

edgeLengthsToTimes If true, the spacing between parent and child nodes will be proportional to the logarithmic time interval between them.

path If present, write tree dumps into this directory. Otherwise, the current directory will be used.

All colors and styles are character strings and can be in any format understood by dot. The tree structure will be dumped to file named mergerTreeDump:⟨ID⟩:⟨N⟩.gv where ⟨ID⟩ is the index of the tree and ⟨N⟩ increasing incrementally from 1 each time the same tree is consecutively dumped. These files can be processed using dot. For example

dot -Tps mergerTreeDump:1:1.gv > tree.ps

will create a tree diagram as the PostScript file tree.ps.
Bibliography


2025


2028


Bibliography


2037
Bibliography


Glossary

**AB magnitude** An astronomical magnitude system in which the apparent magnitude is defined as \( m = -2.5 \log_{10} f - 48.60 \) for a flux density, \( f \), measured in ergs per second per square centimeter per hertz. 207, 2042

**ADAF** An advection-dominated accretion flow (ADAF) is a particular solution for an accretion flow around a black hole, star or compact object in which energy liberated by viscous forces is stored within the accretion flow and advected inward to the central object (see Narayan et al. 1998). 2043

**AHF** Amiga’s Halo Finder (AHF) is a software package which identifies dark matter halos in N-body simulations. Full details are given by Gill et al. [2004] and Knollmann and Knebe [2009]. 2043

**backward descendent** The primary progenitor of a node. This type of descendent is usually relevant when building merger trees and should be distinguished from a forward descendent which is relevant when considering how halos and galaxies evolve forward in time. 2041

**Bernoulli distribution** A discrete probability distribution which takes value 1 with success probability \( p \) and value 0 with failure probability \( q = 1 - p \). Read more on Wikipedia. 766

**BIE** The Bayesian Inference Engine (BIE) is a software package designed to facilitate exploration of complexes parameter spaces using Bayesian techniques. 2043

**CDM** Cold dark matter (CDM) is a hypothesized type of dark matter in which the particles move slowly compared to the speed of light. 2043

**component** An individual physical system within a node, such as a dark matter halo, a galactic disk or a supermassive black hole. 8, 210, 213, 215–218, 221, 222, 224, 228–231, 234, 236–239, 270, 272–275, 296, 307, 2041

**DSL** Domain-specific languages (DSL) are a type of programming language dedicated to a particular problem. In GALACTICUS a DSL is used to specify the structure of components. 2043

**forest** A collection of merger trees that are linked together by virtue of nodes which jump between trees. 151

**forward descendent** The node with which the mass (or majority of the mass) of a node will become associated with at a later time. This type of descendent is usually relevant when considering how halos and galaxies evolve forward in time and should be distinguished from a backward descendent which is relevant when building merger trees. 311, 2041

**HOD** A halo occupation distribution (HOD) is a mathematical model describing the distribution of the number of galaxies (of some given physical properties) found in a dark matter halo of given mass.. 2043

**Lyman continuum** The part of the electromagnetic spectrum which is capable of ionizing hydrogen, i.e. photons with wavelengths shorter than 91.1267 nanometres and with energy above 13.6 eV. 142
**maggie** A unit of luminosity defined to be equal to the luminosity of a zeroth magnitude object in the AB magnitude system. 143, 168

**MD5 hash** The MD5 Message-Digest Algorithm is a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value. In GALACTICUS it is used to encode unique labels for modules which are incorporated into file names. GALACTICUS uses the glibc crypt() function to compute MD5 hashes, but switches “/” for “@” in the hash (since “/” is inconvenient for use in file names). 332

**mergee** For a given node in a merger tree, the set of mergee nodes consists of all nodes which will undergo a galaxy merger with the node at some point in the future. 174

**Millennium Simulation** The Millennium Simulation is a high-resolution N-body simulation of structure formation in a cold dark matter universe. 181


**PAH** Polycyclic aromatic hydrocarbons (PAH) are large organic molecules consisting of fused aromatic rings. 2043

**parent** In a merger tree, the parent node of any given node that exists at time $t_0$ is that node to which it is directly connected in the tree at time $t_1 > t_0$. 173, 174

**primary progenitor** The progenitor of a given node which is regarded as the direct descendent of that node (often, but not always, the most massive progenitor). Other progenitors are considered to merge into this primary progenitor. 173, 174, 2041

**SAM** Semi-analytic models (SAMs) are a type of galaxy formation model utilizing simple parameterizations of physical processes to follow the evolution of galaxies through a merging hierarchy of galaxies. 2043

**UUID** A universally unique identifier—this is a label which uniquely identifies some object (in this case, a GALACTICUS model). 147

**WDM** Warm dark matter (WDM) is a hypothesized type of dark matter in which the particle has non-negligible thermal velocity at decoupling. 2043
Acronyms

**ADAF**  advection-dominated accretion flow. 242, 243, *Glossary: ADAF*

**AHF**  Amiga’s Halo Finder. 288, 289, *Glossary: AHF*

**BIE**  semi-analytic model. 192, 194, 197, 199, 200, *Glossary: BIE*

**CDM**  cold dark matter. 3, 246, 848, 1880, 1881, 1884, *Glossary: CDM*

**CMB**  cosmic microwave background. 37, 251, 770, 1002, 1003

**DSL**  domain-specific language. 8, *Glossary: DSL*

**HOD**  semi-analytic model. 192, 194, 195, *Glossary: HOD*


**IMF**  initial mass function. 37, 73–76, 163, 278–280, 297, 801, 803–805, 838, 843, 858, 859, 1787–1789, 1798

**ISCO**  innermost stable circular orbit. 242, 243

**ISM**  interstellar medium. 210, 299, 884, 885, 1788, 1789

**MCMC**  Markov Chain Monte Carlo. 199, 200

**NFW**  Navarro-Frenk-White (dark matter halo profile). 130, 249, 257, 260, 275, 874

**ODE**  ordinary differential equation. 173, 715

**PAH**  polycyclic aromatic hydrocarbon. 165, *Glossary: PAH*

**SAM**  semi-analytic model. 308, *Glossary: SAM*

**SDSS**  Sloan Digital Sky Survey. 191, 195, 196, 1117

**SED**  spectral energy distribution. 146, 164, 165

**SNe**  supernovae. 59, 60, 112–116, 163

**WDM**  warm dark matter. 1057, 1062, 1882, *Glossary: WDM*
Index

mergerTreeWeight, 144

accretion
  baryonic, 241
  black holes, 212, 214
  disk, 242
accretion disks, 242
active galactic nuclei (AGN)
  feedback, 210, 213
AGN, 142
analysis
  analysis
galacticus, 147
  meta, 147
bar instability, 262
bias
  halo, 249
black holes
  accretion, 212, 214, 311
  jets, 311
  supermassive, 311
C (language) coding, 332
calculation reset task, 332
central galaxies
  identifying, 316
chemical state, 281
chemicals
  reaction rates, 290
clustering
  halo model, 143, 312
code
  directives, 711
coding
  mixed language, 332
components, 712
  creation, 717
  destruction, 717
evolution, 718
implementing, 713
initialization, 717
methods, 718
structure, 712
conditional mass function, 250
Condor, 24
configuration, 21
continuum radiation
  Lyman continuum, 142
cooling, 250, 311
  cooling function, 250
  freefall radius, 254
  infall radius, 254
  output, 311
  radius, 253, 311
  rate, 252, 311
  rate modifier, 253
  specific angular momentum, 254
  time, 255
time available, 255
time available for freefall, 256
cooling function, 250
cooling radius, 253
cooling time, 255
cosmology, 241
CPU utilization
  low, 171
critical density
  mass scaling, 247
dark energy, 242
dark matter, 243
dark matter halo
  concentration, 153
  mass accretion history, 256, 783
  mass loss, 261, 786
  scale radius, 153
  spin, 154
distribution, 262, 785
dark matter halos
  bias, 249
  mass function, 249
dark matter profile
  concentration, 257
  shape, 260
debugging
OpenMP, 24
restoring merger tree internal state, 285
density
  contrast, 311
critical, 247
  virial, 248
density contrast, 311
density profile
  Einasto, 257, 260, 783
  hot halo
    core radius, 276
  Navarro-Frenk-White, 257, 783
dependencies, computation, 331
descendent node, 311
directives, 711
disk
  star formation rate
    output, 222
disks
  stability, 262
dust
  attenuation, 164
  emission, 164
  extinction, 141
  reprocessing, 145
dynamical friction
  satellites, 291
Einasto profile, 257, 260, 783
emission lines, 143, 168
enumerations
  tfamily adjustElements, 334
tfamily componentType, 334
tfamily coordinateSystem, 334
tfamily dataType, 334
tfamily extrapolationType, 335
tfamily massDistributionSymmetry, 335
tfamily massType, 335
tfamily metaDataType, 335
tfamily metallicityScale, 335
tfamily propertyType, 336
tfamily rangeExpand, 336
tfamily rangeExpandSignExpect, 337
tfamily units, 337
tfamily weightBy, 337
evolution, 718
  interrupt, 720
  main branch, 315
excursion sets, 31
feedback, 303
  AGN, 210, 213
    expulsive, 304, 830
filtering
  output, 864
filters
  broadband, 149
flux
  sub-mm, 142
freefall radius, 254
galactic structure, 269
  initial radii, 270
galacticusConfig.xml, 21
galaxies
  central, 316
  indices, 316
  satellite, 316
  tracing through outputs, 316
  weighting, 144
galaxy
  luminosities, 166
  merging, 271
GRAPHViz, 2023
GRASIL, 145, 164
gravitational lensing, 143
half-light radius, 312
half-mass radius, 312
halo bias, 249
halo mass function, 249
halo model, 143, 312
halos
  host mass, 140
heating
  hot halo, 216
histograms, 148
history
  global, 135
hot halo
  density profile
    core radius, 276
  heating, 216
  mass distribution, 275
  ram pressure force, 276
  ram pressure stripping, 277
  timescale, 277
  temperature profile, 277
hydrogen
  chemical, 291
indices
  galaxies, 316
  nodes, 316
infall radius, 254
initial mass function, 278
  selection, 278
instantaneous recycling approximation, 163
intergalactic medium, 280
interrupts, 720
labels, unique, 331
lensing
  gravitational, 143
lightcone, 162
lightcones, 313
linear growth, 246
lines
  emission, 143, 168
load average, 31
luminosities
  galactic, 166
  stellar, 166
Lyman continuum, 142
main branch
  evolution, 315
mass accretion history
  dark matter halo, 256, 783
mass distribution profile
  hot halo, 275
mass loss
  dark matter halo, 261, 786
  ram pressure induced, 291
  tidal force induced, 307
mass profile, 316
  half-light radius, 312
  half-mass radius, 312
merger times, 154
merger tree
  fixed time steps, 289
  links, 316
  mass accretion history, 290
    monotonic, 289
  N-body, 290
  pruning
    by essential node, 290
    by hierarchy, 290
    by mass, 290
  regrid times, 289
  rendering, 316
structure, 318
merger trees, 282
  branching, 286
  building, 287, 2015
  exporting, 2019
  file formats, 2018
  graphing, 2023
  importing, 287
  main branch, 316
  mass resolution, 285
N-body, 155
merging
  galaxy, 271
  progenitor properties, 274
  remnant size, 272
  substructure, 301
  timescale, 301
metadata, 147
migration, 168
Millennium Simulation, 162
mixed language coding, 332
mock catalog, 162
N-body
  merger trees, 155
Navarro-Frenk-White profile, 257, 783
node
  descendent, 311
nodes
  host mass, 140
  indices, 316
  isolated, 316
  substructure, 316
numerical algorithms
  root finding, 329
OpenMP, 13
  debugging, 24
optimization, 332
orbits
  N-body, 157
  satellite
    outputting, 319
  setting, 157
  virial, 157
output
  filtering, 864
  redshift, 137
  temporary, 23
output groups, 865
outputs
  global history, 135  
  rotation curve, 318  
  star formation rate, 319  
  velocity dispersion, 319

parallel, 13
parameters, 168
  generating, 22  
  validating, 22
path
  Galacticus root, 23
plotting, 2021
Population III
  supernovae, 300
power spectrum
  non-linear, 245, 813
  outputting, 34
  primordial, 244
  variance, 244
  window function, 244
pre-derivative task, 865
radiation, 819
ram pressure
  mass loss rate, 291
ram pressure force
  hot halo, 276
ram pressure stripping
  hot halo, 277
reaction rates
  chemical, 290
recycling
  instantaneous, 163
redshift
  output, 137
reionization, 169
root finding, 329
rotation curve
  outputting, 318
run time
  tree construction, 147
  tree evolution, 147
samples
  volume limited, 144
satellite
  merger times, 154
  orbit
    outputting, 319
satellite galaxies
  identifying, 316
satellites
  host mass, 140
  tidal fields, 292
spheroid
  standard
    pseudo-angular momentum, 223
    radius, 223
  star formation rate
    output, 225
spin
  dark matter halo, 262, 785
star formation
  Blitz-Rosolowsky rule, 294
  extended Schmidt law, 294
  Kennicutt-Schmidt law, 294
  Krumholz-McKee-Tumlinson method, 295
  rate
    surface density, 293
    timescale, 295
star formation history, 868
  outputting, 869
  recording, 869
star formation rate
  output
    disk, 222
    spheroid, 225
  outputting, 319
  peak, 143
statistics
  histograms, 148
  Perl modules, 148
stellar
  luminosities, 166
  stellar populations, 296
structure formation, 243
substructure, 301
supermassive black holes
  binary separation, 305
  mergers, 307
  recoil velocity, 306
  separation growth rate, 305
supernovae
  feedback, 303, 304
  Population III, 300
Type Ia, 300
surveys
  geometry, 308
INDEX

task, calculation reset, 332
task, pre-derivative, 865
tasks
  output, 865
temperature profile
  hot halo, 277
tidal fields
  satellites, 292
tidal forces
  mass loss rate, 307
tidal heating
  satellites, 293
tidal stripping
  satellites, 292
timesteps
  criteria, 173
  transfer function, 246
    warm dark matter, 246
unique ID, 332
unique labels, 331
units, 136
velocity dispersion
  outputting, 319
  virial, 320
warm dark matter
  transfer function, 246
Code Index
a1 (function), 931
a2 (function), 931
abundance_pattern_lookup (function), 914
abundances (type), 1325
abundances_add (function), 1325
abundances_allocate_elemental_values (subroutine), 1325
abundances_atomic_index (function), 1325
abundances_builder (subroutine), 1325
abundances_deserialize (subroutine), 1325
abundances_destroy (subroutine), 1325
abundances_divide (function), 1326
abundances_dump (subroutine), 1326
abundances_dump_raw (subroutine), 1326
abundances_get_metallicity (function), 1326
abundances_helium_mass_fraction (function), 1326
abundances_helium_number_fraction (function), 1326
abundances_hydrogen_mass_fraction (function), 1326
abundances_hydrogen_number_fraction (function), 1326
abundances_increment (subroutine), 1326
abundances_initialize (subroutine), 1327
abundances_is_zero (function), 1327
abundances_mass_to_mass_fraction (subroutine), 1327
abundances_mass_to_mass_fraction_packed (subroutine), 1327
abundances_max (function), 1327
abundances_multiply (function), 1327
abundances_multiply_switched (function), 1327
abundances_names (function), 1327
abundances_output (subroutine), 1327
abundances_output_count (subroutine), 1328
abundances_output_names (subroutine), 1328
abundances_property_count (function), 1328
abundances_read_raw (subroutine), 1328
abundances_reset (subroutine), 1328
abundances_serialize (subroutine), 1328
abundances_set_metallicity (subroutine), 1328
abundances_set_to_unity (subroutine), 1328
abundances_structure (module), 1322
abundances_subtract (function), 1328
accretion.Bondi_Hoyle_Lyttleton.F90 (file), 886
accretion.halo.cold_mode.F90 (file), 897
accretion.halo.F90 (file), 886
accretion.halo.Naoz_Barkana_2007.F90 (file), 896
accretion.halo.null.F90 (file), 899
accretion.halo.simple.F90 (file), 900
accretion_disk_adaf_tabulate (subroutine), 902
accretion_disk_jet_power (function), 906
accretion_disk_jet_power_adaf (function), 903
accretion_disk_jet_power_eddington (function), 905
accretion_disk_jet_power_shakura_sunyaev (function), 907
accretion_disk_jet_power_switched (function), 911
accretion_disk_radiative_efficiency (function), 906
accretion_disk_radiative_efficiency_adaf (function), 903
accretion_disk_radiative_efficiency_eddington (function), 905
accretion_disk_radiative_efficiency_shakura_sunyaev (function), 907
accretion_disk_radiative_efficiency_switched (function), 911
accretion_disk_spectra (module), 908
accretion_disk_switched_adaf_fraction (function), 911
accretion_disk_switched_adaf_radiative_efficiency_scaling (function), 911
accretion_disks (module), 906
accretion_disks.ADAF.F90 (file), 902
accretion_disks.Eddington_limited.F90 (file), 905
accretion_disks.F90 (file), 906
accretion_disks.Shakura_Sunyaev.F90 (file), 907
accretion_disks.spectra.F90 (file), 908
accretion_disks.spectra.file.F90 (file), 910
accretion_disks.spectra.Hopkins2007.F90 (file), 910
accretion_disks.switched.F90 (file), 911
accretion_disks.adaf (module), 902
accretion_disks_adaf_get_parameters (subroutine), 903
accretion_disks_adaf_initialize (subroutine), 903
accretion_disks_eddington (module), 905
accretion_disks_eddington_initialize (subroutine), 906
accretion_disks_initialize (subroutine), 906
accretion_disks_shakura_sunyaev (module), 907
accretion_disks_shakura_sunyaev_initialize (subroutine), 907
accretion_disks_switched (module), 911
accretion_disks_switched_initialize (subroutine), 912
accretion_halos (module), 886
accretion_halos_hot_halo_output (subroutine), 888
accretion_halos_hot_halo_output_count (subroutine), 888
accretion_halos_hot_halo_output_names (subroutine), 888
accretion_halos_output_initialize (subroutine), 888
accretiondiskspectra (interface), 908
accretiondiskspectra_class (type), 908
accretiondiskspectra_constructor_default (function), 908
accretiondiskspectra_constructor_named (function), 908
accretiondiskspectra_file (interface), 908, 910
accretiondiskspectra_hopkins2007 (interface), 908, 910
accretiondiskspectra_initialize (subroutine), 908
accretiondiskspectra_is_finalizable (function), 909
accretiondiskspectra_spectrum_null (function), 909
accretionhalo (interface), 888
accretionhalo_accreted_mass_chemicals_null (function), 888
accretionhalo_accreted_mass_metals_null (function), 888
accretionhalo_accreted_mass_null (function), 888
accretionhalo_accretion_rate_chemicals_null (function), 889
accretionhalo_accretion_rate_metals_null (function), 889
accretionhalo_accretion_rate_null (function), 889
accretionhaloclass (type), 889
accretionhalocoldmode (interface), 889, 897
accretionhaloconstructordefault (function), 889
accretionhaloconstructornamed (function), 889
accretionhalofailedaccretedmassnull (function), 889
accretionhalofailedaccretionratenull (function), 889
accretionhaloinitialize (subroutine), 890
accretionhaloisfinalizable (function), 890
accretionhalonaobarkana2007 (interface), 890, 896
accretionhalonull (interface), 890, 899
accretionhalosimple (interface), 890, 900
adaf_alpha (function), 903
adaf_angular_momentum (function), 903
adaf_bh_jet_power (function), 903
adaf_disk_jet_power (function), 903
adaf_disk_jet_power_from_black_hole (function), 904
adaf_enthalpy (function), 904
adaf_enthalpy_angular_momentum_product (function), 904
adaf_field_enhancement (function), 904
adaf_fluid_angular_velocity (function), 904
adaf_gamma (function), 904
adaf_gamma_phi (function), 904
adaf_gamma_r (function), 904
adaf_height (function), 904
adaf_temperature (function), 905
adaf_v (function), 905
add_memory_component (subroutine), 1968
adiabatic_solver_mean_orbital_radius (function), 1078
adiabatic_solver_mean_orbital_radius_derivative (function), 1078
adjustl (interface), 1214
adjustl_ (function), 1214
adjustr (interface), 1214
adjustr_ (function), 1214
agestatisticscountlinked (function), 1374
agestatisticscreatebyinterrupt (subroutine), 1374
agestatisticscreatelinked (subroutine), 1374
agestatisticsdestroylinked (subroutine), 1374
agestatisticsdiskintegratedsfr (function), 1374
agestatisticsdiskintegratedsfrattributematch (function), 1375
agestatisticsdiskintegratedsfrisgettable (function), 1375
agestatisticsdiskintegratedsfrrate (subroutine), 1375
agestatisticsdisktimeweightedintegratedsfr (function), 1375
agestatisticsdisktimeweightedintegratedsfrattributematch (function), 1375
agestatisticsdisktimeweightedintegratedsfrisgettable (function), 1375
agestatisticsdisktimeweightedintegratedsfrrate (subroutine), 1375
agestatisticsget (function), 1375
agestatisticsnullbindingdouble0inout (function), 1376
agestatisticsnullbindinginteger0in (function), 1376
agestatisticsnullbindingratedouble0inout (subroutine), 1376
agestatisticsnullbindingrateinteger0in (subroutine), 1376
agestatisticsnullbindingsetdouble0inout (subroutine), 1376
agestatisticsnullbindingsetinteger0in (subroutine), 1376
agestatisticsspheroidintegratedsfr (function), 1376
agestatisticsspheroidintegratedsfrattributematch (function), 1376
agestatisticsspheroidintegratedsfrisgettable (function), 1376
agestatisticsspheroidintegratedsfrrate (subroutine), 1377
agestatisticsspheroidtimeweightedintegratedsfr (function), 1377
agestatisticsspheroidtimeweightedintegratedsfrattributematch (function), 1377
agestatisticsspheroidtimeweightedintegratedsfrisgettable (function), 1377
agestatisticsspheroidtimeweightedintegratedsfrrate (subroutine), 1377
agestatisticsstandarddiskintegratedsfrcount (function), 1377
agestatisticsstandarddiskintegratedsfrget (function), 1377
agestatisticsstandarddiskintegratedsfrrate (subroutine), 1378
agestatisticsstandarddiskintegratedsfrscale (subroutine), 1378
agestatisticsstandarddiskintegratedsfrset (subroutine), 1378
agestatisticsstandarddisktimeweightedintegratedsfrcount (function), 1378
agestatisticsstandarddisktimeweightedintegratedsfrget (function), 1378
agestatisticsstandarddisktimeweightedintegratedsfrrate (subroutine), 1378
agestatisticsstandarddisktimeweightedintegratedsfrscale (subroutine), 1378
agestatisticsstandarddisktimeweightedintegratedsfrset (subroutine), 1378
agestatisticsstandardspheroidintegratedsfrcount (function), 1379
agestatisticsstandardspheroidintegratedsfrget (function), 1379
agestatisticsstandardspheroidintegratedsfrrate (subroutine), 1379
agestatisticsstandardspheroidintegratedsfrscale (subroutine), 1379
agestatisticsstandardspheroidintegratedsfrset (subroutine), 1379
agestatisticsstandardspheroidtimeweightedintegratedsfrcount (function), 1379
agestatisticsstandardspheroidtimeweightedintegratedsfrget (function), 1379
agestatisticsstandardspheroidtimeweightedintegratedsfrrate (subroutine), 1379
agestatisticsstandardspheroidtimeweightedintegratedsfrscale (subroutine), 1380
agestatisticsstandardspheroidtimeweightedintegratedsfrset (subroutine), 1380
alfalfa_hi_mass_function_z0_00_initialize (subroutine), 1119
alloc_array (interface), 1968
alloc_array_character_1d (subroutine), 1968
alloc_array_character_1d_kind_int8 (subroutine), 1968
alloc_array_double_precision_1d (subroutine), 1969
alloc_array_double_precision_1d_kind_int8 (subroutine), 1969
alloc_array_double_precision_2d (subroutine), 1969
alloc_array_double_precision_2d_kind_int8 (subroutine), 1969
alloc_array_double_precision_3d (subroutine), 1969
alloc_array_double_precision_3d_kind_int8 (subroutine), 1969
alloc_array_double_precision_4d (subroutine), 1969
alloc_array_double_precision_4d_kind_int8 (subroutine), 1970
alloc_array_double_precision_5d (subroutine), 1970
alloc_array_double_precision_5d_kind_int8 (subroutine), 1970
alloc_array_integer_1d (subroutine), 1970
alloc_array_integer_1d_kind_int8 (subroutine), 1970
alloc_array_integer_2d (subroutine), 1970
alloc_array_integer_2d_kind_int8 (subroutine), 1970
CODE INDEX

alloc_array_integer_kind_int8_1d (subroutine), 1971
alloc_array_integer_kind_int8_1d_kind_int8 (subroutine), 1971
alloc_array_integer_kind_int8_2d (subroutine), 1971
alloc_array_integer_kind_int8_2d_kind_int8 (subroutine), 1971
alloc_array_logical_1d (subroutine), 1971
alloc_array_logical_1d_kind_int8 (subroutine), 1971
alloc_array_real_1d (subroutine), 1971
alloc_array_real_1d_kind_int8 (subroutine), 1972
allocate_buffers (subroutine), 1121
amaximumroot (function), 1878
array_cumulate (interface), 1931
array_cumulate_double (function), 1931
array_index (interface), 1931
array_index_double (function), 1931
array_index_double_2d (function), 1931
array_index_integer (function), 1931
array_index_integer8 (function), 1931
array_intersection_varying_string (function), 1932
array_is_monotonic (interface), 1932
array_is_monotonic_double (function), 1932
array_is_monotonic_integer8 (function), 1932
array_reverse (interface), 1932
array_reverse_double (function), 1932
array_reverse_real (function), 1932
array_reverse_sizet (function), 1932
array_utilities (module), 1930
array_which (subroutine), 1932
arrays_search (module), 1319
assert (interface), 1984
assert_character_1d_array (subroutine), 1984
assert_character_scalar (subroutine), 1984
assert_double_1d_array (subroutine), 1984
assert_double_2d_array (subroutine), 1984
assert_double_3d_array (subroutine), 1984
assert_double_4d_array (subroutine), 1984
assert_double_5d_array (subroutine), 1985
assert_double_scalar (subroutine), 1985
assert_integer8_1d_array (subroutine), 1985
assert_integer8_scalar (subroutine), 1985
assert_integer_1d_array (subroutine), 1985
assert_integer_scalar (subroutine), 1985
assert_logical_1d_array (subroutine), 1985
assert_logical_scalar (subroutine), 1985
assert_real_1d_array (subroutine), 1986
assert_real_scalar (subroutine), 1986
assert_varstring_1d_array (subroutine), 1986
assert_varstring_scalar (subroutine), 1986
assertresult (type), 1986
assign_isolated_node_indices (subroutine), 1245
assign_mergers (subroutine), 1246
assign_particle_counts (subroutine), 1246
assign_scale_radii (subroutine), 1246
assign_spin_parameters (subroutine), 1246
assign_uniqueids_to_clones (subroutine), 1246
assign_velocity_dispersions (subroutine), 1246
assign_velocity_maxima (subroutine), 1246
assignment(=) (interface), 890, 896, 897, 1214, 1335, 1380, 1735
atom_lookup (function), 914
atomic.cross_sections.Compton.F90 (file), 912
atomic.cross_sections.ionization.photo.F90 (file), 912
atomic.cross_sections.ionization.photo.Verner.F90 (file), 913
atomic.data.F90 (file), 914
atomic.ionization_potentials.F90 (file), 915
atomic.ionization_potentials.Verner.F90 (file), 915
atomic.radiation.gaunt_factors.F90 (file), 916
atomic.radiation.gaunt_factors.Sutherland1998.F90 (file), 916
atomic.rates.excitation.collisional.F90 (file), 917
atomic.rates.excitation.collisional.ScholzWalters91.F90 (file), 917
atomic.rates.ionization.collisional.F90 (file), 918
atomic.rates.ionization.collisional.Verner.F90 (file), 918
atomic.rates.recombination.dielectronic.Arnaud85.F90 (file), 919
atomic.rates.recombination.dielectronic.F90 (file), 921
atomic.rates.recombination.radiative.data.F90 (file), 923
atomic.rates.recombination.radiative.F90 (file), 921
atomic.rates.recombination.radiative.Verner.F90 (file), 922
atomic_abundance (function), 914
atomic_cross_section_compton (function), 912
atomic_cross_section_ionization_photo (function), 912
atomic_cross_section_ionization_photo_initialize (subroutine), 913
atomic_cross_section_ionization_photo_verner (function), 913
atomic_cross_section_ionization_photo_verner_initialize (subroutine), 913
atomic_cross_sections_compton (module), 912
atomic_cross_sections.ionization.photo (module), 912
atomic_cross_sections.ionization.photo_verner (module), 913
atomic_data (module), 914
atomic_data_atoms_count (function), 914
atomic_data_initialize (subroutine), 914
atomic_ionization_potential (function), 915
atomic_ionization_potential_initialize (subroutine), 915
atomic_ionization_potential_verner (function), 916
atomic_ionization_potentials (module), 915
atomic_ionization_potentials_verner (module), 915
atomic_ionization_potentials_verner_initialize (subroutine), 916
atomic_mass (function), 915
atomic_radiationgaunt_factors (module), 916
atomic_rate_ionization_collisional (function), 918
atomic_rate_ionization_collisional_initialize (subroutine), 918
atomic_rate_ionization_collisional_verner (function), 919
atomic_rate_ionization_collisional_verner_initialize (subroutine), 919
atomic_rate_recombination_radiative (function), 921
atomic_rate_recombination_radiative_initialize (subroutine), 921
atomic_rate_recombination_radiative_verner (function), 922
atomic_rate_recombination_radiative_verner_initialize (subroutine), 922
atomic_rates_excitation_collisional (module), 917
atomic_rates_excitation_collisional_scholzwalter91 (module), 918
atomic_rates Ionization_collisional (module), 918
atomic_rates Ionization_collisional_verner (module), 919
atomic_rates_recombination_dielectronic (module), 921
atomic_rates_recombination_dielectronic_arnaud1985 (module), 919
atomic_rates_recombination_radiative (module), 921
atomic_rates_recombination_radiative_data (module), 923
atomic_rates_recombination_radiative_verner (module), 922
atomic_short_label (function), 915
atomicbond (type), 1333
atomicdata (type), 915
atomicstructure (type), 1333

backgroundradiationodes (function), 1743
bar_instability_timescale (subroutine), 1069
bar_instability_timescale_eln (subroutine), 1068
bar_instability_timescale_eln_tidal (subroutine), 1068
bar_instability_timescale_null (subroutine), 1069
basicaccretionrate (function), 1380
basicaccretionrateattributematch (function), 1380
basicaccretionrateisgettable (function), 1380
basiccountlinked (function), 1380
basiccreatebyinterrupt (subroutine), 1380
basiccreatelinked (subroutine), 1380
basicdestroylinked (subroutine), 1381
basicget (function), 1381
basicmass (function), 1381
basicmassattributematch (function), 1381
basicmassisgettable (function), 1381
basicmassmaximum (function), 1381
basicmassmaximumattributematch (function), 1381
basicmassmaximumisgettable (function), 1381
basicnonenvolvingmassget (function), 1381
basicnonenvovingmassset (subroutine), 1382
basicnonenvolvingtimecount (function), 1382
basicnonenvolvingtimeget (function), 1382
basicnonenvovingtimelastisolated (function), 1382
basicnonenvovingtimelastisolatedset (subroutine), 1382
basicnonenvolvingtimerate (subroutine), 1382
basicnonenvolvingtimesc (subroutine), 1382
basicnonenvolvingtimeset (subroutine), 1382
basicnullbindingdouble0inout (function), 1382
basicnullbindinginteger0in (function), 1383
basicnullbindingratedouble0inout (subroutine), 1383
basicnullbindingrateinteger0in (subroutine), 1383
basicnullbindingsetdouble0inout (subroutine), 1383
basicnullbindingsetinteger0in (subroutine), 1383
basicstandardaccretionrateget (function), 1383
basicstandardaccretionrateset (subroutine), 1383
basicstandardmasscount (function), 1383
basicstandardmassget (function), 1383
basicstandardmassrate (subroutine), 1384
basicstandardmassscale (subroutine), 1384
basicstandardmassset (subroutine), 1384
basicstandardtimeoutcount (function), 1384
basicstandardtimeoutget (function), 1384
basicstandardtimelastisolated (function), 1384
basicstandardtimelastisolatedset (subroutine), 1384
basicstandardtimescale (subroutine), 1384
basicstandardtimeset (subroutine), 1385
basicstandardtrackingmassmaximumget (function), 1385
basicstandardtrackingmassmaximumset (subroutine), 1385
basictime (function), 1385
basictimeattributematch (function), 1385
basictimeisgettable (function), 1385
basictimelastisolated (function), 1385
basictimelastisolatedattributematch (function), 1385
basictimelastisolatedisgettable (function), 1385
bessel_function_i0 (function), 1225
bessel_function_i1 (function), 1225
bessel_function_k0 (function), 1225
bessel_function_k1 (function), 1225
bessel_functions (module), 1225
betaprofileconstructor (function), 1163, 1168
betaprofiledefaultconstructor (function), 1163, 1168
betaprofiledensity (function), 1163, 1168
betaprofiledensitylogslope (function), 1164, 1168
betaprofiledestructor (subroutine), 1164, 1168
betaprofileenclosedmass (function), 1164, 1169
betaprofileinitialize (subroutine), 1164, 1169
betaprofileradialmoment (function), 1164, 1169
betaprofilerotationnormalization (function), 1164, 1169
bias_integrand_i (function), 1818
bivar (module), 923
bivar.F90 (file), 923
black_hole_binary_initial_radii (module), 924
black_hole_binary_initial_radii_spheroid_size (module), 925
black_hole_binary_initial_radii_spheroid_size_initialize (subroutine), 925
black_hole_binary_initial_radii_tidal_radius (module), 925
black_hole_binary_initial_radii_tidal_radius_initialize (subroutine), 926
black_hole_binary_initial_radii_volonteri_2003 (module), 924
black_hole_binary_initial_radii_volonteri_2003_initialize (subroutine), 924
black_hole_binary_initial_radius (function), 924
black_hole_binary_initial_radius_spheroid_size (function), 925
black_hole_binary_initial_radius_tidal_radius (function), 926
black_hole_binary_initial_radius_volonteri_2003 (function), 925
black_hole_binary_merger (subroutine), 930
black_hole_binary_merger_rezzolla (subroutine), 930
black_hole_binary_merger_rezzolla_initialize (subroutine), 930
black_hole_binary_mergers (module), 930
black_hole_binary_mergers_rezzolla (module), 930
black_hole_binary_recoil_velocities (module), 927
black_hole_binary_recoil_velocities_standard (module), 926
black_hole_binary_recoil_velocity (function), 927
black_hole_binary_recoil_velocity_null (function), 928
black_hole_binary_recoil_velocity_null_initialize (subroutine), 928
black_hole_binary_recoil_velocity_standard (function), 926
black_hole_binary_recoil_velocity_standard_initialize (subroutine), 926
black_hole_binary_recoil_velocity_standard_snapshot (subroutine), 927
black_hole_binary_recoil_velocity_standard_state_retrieve (subroutine), 927
black_hole_binary_recoil_velocity_standard_state_store (subroutine), 927
black_hole_binary_separation_growth_rate (function), 928
black_hole_binary_separation_growth_rate_null (function), 929
black_hole_binary_separation_growth_rate_null_initialize (subroutine), 929
black_hole_binary_separation_growth_rate_standard (function), 929
black_hole_binary_separation_growth_rate_standard_init (subroutine), 929
black_hole_binary_separations (module), 928
black_hole_binary_separations_null (module), 928
black_hole_binary_separations_standard (module), 929
black_hole_eddington_accretion_rate (function), 931
black_hole_frame_dragging_frequency (interface), 931
black_hole_frame_dragging_frequency_node (function), 931
black_hole_frame_dragging_frequency_spin (function), 932
black_hole_fundamentals (module), 931
black_hole_gravitational_radius (function), 932
black_hole_horizon_radius (interface), 932
black_hole_horizon_radius_node (function), 932
black_hole_horizon_radius_spin (function), 932
black_hole_isco_radius (interface), 932
black_hole_isco_radius_node (function), 932
black_hole_isco_radius_spin (function), 932
black_hole_isco_specific_angular_momentum (function), 932
black_hole_isco_specific_energy (interface), 933
black_hole_isco_specific_energy_node (function), 933
black_hole_isco_specific_energy_spin (function), 933
black_hole_metric_a_factor (interface), 933
black_hole_metric_a_factor_node (function), 933
black_hole_metric_a_factor_spin (function), 933
black_hole_metric_d_factor (interface), 933
black_hole_metric_d_factor_node (function), 933
black_hole_metric_d_factor_spin (function), 934
black_hole_recoil_velocities_null (module), 927
black_hole_rotational_energy_spin_down (interface), 934
black_hole_rotational_energy_spin_down_node (function), 934
black_hole_rotational_energy_spin_down_spin (function), 934
black_hole_spin_up_rate (function), 907
black_hole_spin_up_rate_adaf (function), 905
black_hole_spin_up_rate_eddington (function), 906
black_hole_spin_up_rate_shakura_sunyaev (function), 907
black_hole_spin_up_rate_switched (function), 912
black_hole_static_radius (interface), 934
black_hole_static_radius_node (function), 934
black_hole_static_radius_spin (function), 934
black_holes.binaries.initial_radius.F90 (file), 924
black_holes.binaries.initial_radius.spheroid_size_fraction.F90 (file), 925
black_holes.binaries.initial_radius.tidal_radius.F90 (file), 925
black_holes.binaries.initial_radius.Volonteri_2003.F90 (file), 924
black_holes.binaries.recoil_velocity.Campanelli2007.F90 (file), 926
black_holes.binaries.recoil_velocity.F90 (file), 927
black_holes.binaries.recoil_velocity.null.F90 (file), 927
black_holes.binaries.separation_growth_rate.F90 (file), 928
black_holes.binaries.separation_growth_rate.null.F90 (file), 928
black_holes.binaries.separation_growth_rate.standard.F90 (file), 929
black_holes.binary_mergers.F90 (file), 930
black_holes.binary_mergers.Rezzolla2008.F90 (file), 930
black_holes.fundamentals.F90 (file), 930
blackbody_emission (function), 1913
blackholeaccretionrate (function), 1386
blackholeaccretionrateattributematch (function), 1386
blackholeaccretionratesisgettable (function), 1386
blackholecountlinked (function), 1386
blackholecreatebyinterrupt (subroutine), 1386
blackholecreatelinked (subroutine), 1386
blackholedeestroylinked (subroutine), 1386
blackholeget (function), 1386
blackholeradialposition (function), 1387
blackholeradialpositionattributematch (function), 1387
blackholeradialpositionisgettable (function), 1387
blackholeradialpositionseed (function), 1387
blackholeradialpositionseedattributematch (function), 1387
blackholeradialpositionseedisgettable (function), 1387
blackholenullbindingdouble0inout (function), 1387
blackholenullbindinginteger0inout (function), 1387
blackholenullbindingratedouble0inout (subroutine), 1387
blackholenullbindingrateinteger0in (subroutine), 1388
blackholenullbindingsdouble0inout (subroutine), 1388
blackholenullbindingsetinteger0in (subroutine), 1388
blackholeradialposition (function), 1388
blackholeradialpositionattributematch (function), 1388
blackholeradialpositionisgettable (function), 1388

2059
blackholeradiativeefficiency (function), 1388
blackholeradiativeefficiencyattributematch (function), 1388
blackholeradiativeefficiencyisgettable (function), 1388
blackholesimplemasscount (function), 1389
blackholesimplemassget (function), 1389
blackholesimplemassrate (subroutine), 1389
blackholesimplemasssscale (subroutine), 1389
blackholesimplemassset (subroutine), 1389
blackholespin (function), 1389
blackholespinattributematch (function), 1389
blackholespinisgettable (function), 1389
blackholespinseed (function), 1389
blackholespinseedattributematch (function), 1390
blackholespinseedisgettable (function), 1390
blackholestandardaccretionrateget (function), 1390
blackholestandardaccretionrategetfunction (subroutine), 1390
blackholestandardaccretionrategetisattached (function), 1390
blackholestandardmasscount (function), 1390
blackholestandardmassget (function), 1390
blackholestandardmassrate (subroutine), 1390
blackholestandardmassset (subroutine), 1390
blackholestandardmasssscale (subroutine), 1391
blackholestandardmassset (subroutine), 1391
blackholestandardradialpositioncount (function), 1391
blackholestandardradialpositionget (function), 1391
blackholestandardradialpositionrate (subroutine), 1391
blackholestandardradialpositionsscale (subroutine), 1391
blackholestandardradialpositionset (subroutine), 1391
blackholestandardradiativeefficiencyget (function), 1391
blackholestandardradiativeefficiencygetfunction (subroutine), 1391
blackholestandardradiativeefficiencygetisattached (function), 1392
blackholestandardspincount (function), 1392
blackholestandardspinrate (subroutine), 1392
blackholestandardspinscale (subroutine), 1392
blackholestandardspinset (subroutine), 1392
blackholestandardtripleinteractiontimeget (function), 1392
blackholestandardtripleinteractiontimeset (subroutine), 1392
blackholetripleinteractiontime (function), 1392
blackholetripleinteractiontimeattributematch (function), 1393
blackholetripleinteractiontimeisgettable (function), 1393
bolshoi_merger_tree_file_maker (program), 881
Bolshoi_Merger_Tree_File_Maker.F90 (file), 881
bondi_hoyle_lyttleton_accretion (module), 886
bondi_hoyle_lyttleton_accretion_radius (function), 886
bondi_hoyle_lyttleton_accretion_rate (function), 886
boolean_false (function), 1393
boolean_true (function), 1393
boylankolchin2008defaultconstructor (function), 1757, 1759
boylankolchin2008destructor (subroutine), 1758, 1760
boylankolchin2008timeuntilmerging (function), 1758, 1760
branching_probability_integrand (function), 1234
branching_probability_integrand_generalized (function), 1232
bryannorman1998defaultconstructor (function), 1885, 1886
bryannorman1998densitycontrast (function), 1885, 1886
bryannorman1998densitycontrastrateofchange (function), 1885, 1886
build_child_and_sibling_links (subroutine), 1246
build_descendent_pointers (subroutine), 1247
build_isolated_parent_pointers (subroutine), 1247
build_parent_pointers (subroutine), 1247
build_subhalo_mass_histories (subroutine), 1247

chabrier_phi (function), 1785
char (interface), 1214
char_auto (function), 1214
char_fixed (function), 1215
chemical.reaction_rates.F90 (file), 935
chemical.reaction_rates.hydrogen.F90 (file), 935
chemical.reaction_rates.null.F90 (file), 941
chemical.reaction_rates.utilities.F90 (file), 941
chemical.state.atomic_cie_cloudy.F90 (file), 945
chemical.state.CIE_file.F90 (file), 942
chemical.state.F90 (file), 944
chemical_abundances_add (function), 1330
chemical_abundances_allocate_values (subroutine), 1330
chemical_abundances_deserialize (subroutine), 1330
chemical_abundances_divide (function), 1330
chemical_abundances_increment (subroutine), 1331
chemical_abundances_initialize (subroutine), 1331
chemical_abundances_multiply (function), 1331
chemical_abundances_serialize (subroutine), 1331
chemical_abundances_structure (module), 1329
chemical_abundances_subtract (function), 1331
chemical_database_get (subroutine), 1333
chemical_database_get_index (function), 1334
chemical_densities (subroutine), 945
chemical_densities_atomic_cie_cloudy (subroutine), 946
chemical_densities_cie_file (subroutine), 942
chemical_densities_cie_file_interpolate (subroutine), 943
chemical_hydrogen_rate_h2_electron_to_2h_electron (subroutine), 935
chemical_hydrogen_rate_h2_gamma_to_2h (subroutine), 935
chemical_hydrogen_rate_h2_gamma_to_h2plus_electron (subroutine), 936
chemical_hydrogen_rate_h2_gamma_to_h2star_to_2h (subroutine), 936
chemical_hydrogen_rate_h2_h_to_3h (subroutine), 936
chemical_hydrogen_rate_h2_hplus_to_h2plus_h (subroutine), 936
chemical_hydrogen_rate_h2plus_electron_to_2h (subroutine), 936
chemical_hydrogen_rate_h2plus_gamma_to_2hplus_electron (subroutine), 936
chemical_hydrogen_rate_h2plus_gamma_to_h_hplus (subroutine), 936
chemical_hydrogen_rate_h2plus_h_to_h2_hplus (subroutine), 937
chemical_hydrogen_rate_h2plus_hminus_to_h2_h (subroutine), 937
chemical_hydrogen_rate_h_electron_to_hminus_photon (subroutine), 937
chemical_hydrogen_rate_h_electron_to_hplus_2electron (subroutine), 937
chemical_hydrogen_rate_h_gamma_to_hplus_electron (subroutine), 937
chemical_hydrogen_rate_h_hminus_to_h2_electron (subroutine), 937
chemical_hydrogen_rate_h_hplus_to_h2plus_photon (subroutine), 937
chemical_hydrogen_rate_hminus_electron_to_h_2electron (subroutine), 938
chemical_hydrogen_rate_hminus_gamma_to_h_electron (subroutine), 938
chemical_hydrogen_rate_hminus_h_to_2h_electron (subroutine), 938
chemical_hydrogen_rate_hminus_hplus_to_2h (subroutine), 938
chemical_hydrogen_rate_hminus_hplus_to_h2plus_electron (subroutine), 938
chemical_hydrogen_rate_hplus_electron_to_h_photon (subroutine), 938
chemical_hydrogen_rates (module), 935
chemical_hydrogen_rates_compute (subroutine), 938
chemical_hydrogen_rates_initialize (subroutine), 938
chemical_reaction_rate (subroutine), 935
chemical_reaction_rates (module), 935
chemical_reaction_rates_initialize (subroutine), 938
chemical_reaction_rates_null (module), 941
chemical_reaction_rates_null_compute (subroutine), 941
chemical_reaction_rates_null_initialize (subroutine), 941
chemical_reaction_rates_utilities (module), 942
chemical_state_atomic_cie_cloudy_create (subroutine), 946
chemical_state_atomic_cie_cloudy_initialize (subroutine), 946
chemical_state_cie_chemicals_initialize (subroutine), 943
chemical_state_cie_file_format_version (function), 943
chemical_state_cie_file_initialize (subroutine), 943
chemical_state_cie_file_read (subroutine), 943
chemical_state_cie_file_read_initialize (subroutine), 943
chemical_state_initialize (subroutine), 945
chemical_states (module), 944
chemical_states_atomic_cie_cloudy (module), 946
chemical_states_cie_file (module), 942
chemical_structure_charge (function), 1334
chemical_structure_export (subroutine), 1334
chemical_structure_initialize (subroutine), 1334
chemical_structure_mass (function), 1334
chemical_structures (module), 1333
chemicalabundances (type), 1331
chemicals_abundances (function), 1331
chemicals_abundances_destroy (subroutine), 1331
chemicals_abundances_is_zero (function), 1331
chemicals_abundances_reset (subroutine), 1332
chemicals_abundances_set (subroutine), 1332
chemicals_abundances_set_to_unity (subroutine), 1332
chemicals_builder (subroutine), 1332
chemicals_dump (subroutine), 1332
chemicals_dump_raw (subroutine), 1332
chemicals_enforce_positive (subroutine), 1332
chemicals_index (function), 1332
chemicals_mass_to_density_conversion (function), 942
chemicals_mass_to_number (subroutine), 1332
chemicals_names (function), 1333
chemicals_number_to_mass (subroutine), 1333
chemicals_property_count (function), 1333
chemicals_read_raw (subroutine), 1333
chemicalstructure (type), 1334
circularity_cumulative_probability (function), 1774
circularity_root (function), 1775
close_parameters_group (subroutine), 1948
cnode_component_agemassstatistics_diskintegratedsfr (function), 1671
cnode_component_agemassstatistics_disktimeweightedintegratedsfr (function), 1671
cnode_component_agemassstatistics_spheroidintegratedsfr (function), 1671
cnode_component_agemassstatistics_spheroidtimeweightedintegratedsfr (function), 1672

cnode_component_basic_accretionrate (function), 1672
cnode_component_basic_mass (function), 1672
cnode_component_basic_massmaximum (function), 1672
cnode_component_basic_time (function), 1672
cnode_component_blackhole_accretionrate (function), 1672

cnode_component_blackhole_radialposition (function), 1672
cnode_component_blackhole_radialvelocity (function), 1673
cnode_component_blackhole_spin (function), 1673
cnode_component_blackhole_spinseed (function), 1673

cnode_component_hothalo_angularmomentum (function), 1673
cnode_component_hothalo_angularmomentumcold (function), 1674
cnode_component_hothalo_angularmomentumhot (function), 1674

cnode_component_disk_starformationrate (function), 1674
cnode_component_disk_velocity (function), 1674

cnode_component_formattime (function), 1674

cnode_component_hothalo_mass (function), 1674

cnode_component_hothalo_masscold (function), 1675
cnode_component_hothalo_outerradius (function), 1675

cnode_component_hothalo_unaccretedmass (function), 1675
cnode_component_interoutput_diskstarformationrate (function), 1675

cnode_component_interoutput_spheroidstarformationrate (function), 1675

cnode_component_massflowstatistics_cooledmass (function), 1676

cnode_component_mergingstatistics_galaxymajormergertime (function), 1676

cnode_component_mergingstatistics_nodeformationtime (function), 1676

cnode_component_mergingstatistics_nodemajormergertime (function), 1676

cnode_component_nbody_velocitydispersion (function), 1676

cnode_component_nbody_velocitymaximum (function), 1676

cnode_component_satellite_boundmass (function), 1676

cnode_component_satellite_mergetime (function), 1676

cnode_component_satellite_tidalheatingnormalized (function), 1676

cnode_component_satellite_timeofmerging (function), 1676

cnode_component_spheroid_angularmomentum (function), 1677

cnode_component_spheroid_halfmassradius (function), 1677

cnode_component_spheroid_massgas (function), 1677

cnode_component_spheroid_massstellar (function), 1677

cnode_component_spheroid_radius (function), 1677

cnode_component_spheroid_starformationrate (function), 1677

cnode_component_spheroid_velocity (function), 1677

cnode_component_spin_spin (function), 1677

cnode_component_spin_spingrowthrate (function), 1677

code_memory_usage (module), 1933

coldmodeaccretedmass (function), 890, 897

coldmodeaccretedmasschemicals (function), 890, 897

coldmodeaccretedmassmetals (function), 891, 897

coldmodeaccretionrate (function), 891, 897

coldmodeaccretionratechemicals (function), 891, 897

coldmodeaccretionratemetals (function), 891, 897

coldmodechemicalmasses (function), 891, 898

coldmodecoldmodefraction (function), 891, 898

coldmodeconstructor (function), 892, 898

coldmodedefaultconstructor (function), 892, 898

coldmodefailedaccretedmass (function), 892, 898

coldmodefailedaccretionrate (function), 892, 898

coldmodefromsimple (subroutine), 892, 899

collapseroot (function), 1879

collapsing_mass_root (function), 1844

collisional_excitation_cooling_rate (function), 917, 918

collisional_excitation_cooling_rate_scholzwalterm91_initialize (subroutine), 918

collisional_excitation_rate_initialize (subroutine), 917

cd_command_arguments (module), 1933

cd_compare_double (function), 1321

cd_compare_integer (function), 1321

cd_compare_integer8 (function), 1321

cd_component_density (function), 1070

cd_component_enclosed_mass (function), 1071, 1079

cd_component_potential (function), 1075

cd_component_rotation_curve (function), 1079, 1083

cd_component_rotation_curve_gradient (function), 1079, 1083
component_surface_density (function), 1084
compute_common_factors (subroutine), 1233, 1235
concatenate_varstr_integer (function), 1981
concatenate_varstr_integer8 (function), 1981
conditional_mass_function (program), 881
Conditional_Mass_Function.F90 (file), 881
conditional_mass_functions (module), 1161
conditional_mass_functions_behroozi2010 (module), 1160
conditional_mass_functions_behroozi2010_initialize (subroutine), 1160
conditional_mass_functions_initialize (subroutine), 1161
constant.F90 (file), 947
constants_nswc (module), 947
convert_varstring_to_char (function), 1981
cooling.cold_mode.infall_rate.dynamical_time.F90 (file), 948
cooling.cold_mode.infall_rate.F90 (file), 947
cooling.cooling_function.atomic_CIE_Cloudy.F90 (file), 953
cooling.cooling_function.CIE_file.F90 (file), 949
cooling.cooling_function.CMB_Compton.F90 (file), 951
cooling.cooling_function.F90 (file), 952
cooling.cooling_function.molecular_hydrogen_Galli_Palla.F90 (file), 954
cooling.cooling_radius.F90 (file), 955
cooling.cooling_radius.isothermal_profile.F90 (file), 957
cooling.cooling_radius.simple.F90 (file), 958
cooling.cooling_rate.F90 (file), 959
cooling.cooling_rate.modifier.cut_off.F90 (file), 961
cooling.cooling_rate.simple.F90 (file), 961
cooling.cooling_rate.simple_scaling.F90 (file), 962
cooling.cooling_rate.White-Frenk.F90 (file), 960
cooling.cooling_rate.zero.F90 (file), 962
cooling.cooling_time.F90 (file), 963
cooling.cooling_time.simple.F90 (file), 964
cooling.freefall_radii.dark_matter_halo.F90 (file), 965
cooling.freefall_radii.F90 (file), 964
cooling.freefall_time_available.F90 (file), 966
cooling.freefall_time_available.halo_formation.F90 (file), 967
cooling.infall_radius.cooling_and_freefall.F90 (file), 968
cooling.infall_radius.cooling_radius.F90 (file), 969
cooling.infall_radius.cooling_radius.F90 (file), 969
cooling.infall_radius.F90 (file), 967
cooling.specific_angular_momentum.constant_rotation.F90 (file), 970
cooling.specific_angular_momentum.F90 (file), 969
cooling.specific_angular_momentum.mean.F90 (file), 970
cooling.time_available.F90 (file), 971
cooling.time_available.halo_formation.F90 (file), 972
cooling.time_available.White-Frenk.F90 (file), 972
cooling_cold_mode_infall_output (subroutine), 948
cooling_cold_mode_infall_output_count (subroutine), 948
cooling_cold_mode_infall_output_initialize (subroutine), 948
cooling_cold_mode_infall_output_names (subroutine), 948
cooling_cold_mode_infall_rate (function), 948
cooling_cold_mode_infall_rate_dynamical_time (function), 949
cooling_cold_mode_infall_rate_dynamical_time_initialize (subroutine), 949
cooling_cold_mode_infall_rate_initialize (subroutine), 948
cooling_cold_mode_infall_rates (module), 947
cooling_cold_mode_infall_rates_dynamical_time (module), 948
cooling_freefall_time_available (function), 966
cooling_freefall_time_available_get_template (function), 966
cooling_freefall_time_available_get_template (interface), 966
cooling_freefall_time_available_increase_rate (function), 966
cooling_freefall_time_available_initialize (subroutine), 966
cooling_freefall_times_available (module), 966
cooling_function (function), 952
cooling_function_atomic_cie_cloudy (subroutine), 953
cooling_function_atomic_cie_cloudy_create (subroutine), 953
cooling_function_atomic_cie_cloudy_initialize (subroutine), 954
cooling_function_cie_file (subroutine), 949
cooling_function_cie_file_format_version (function), 949
cooling_function_cie_file_initialize (subroutine), 949
cooling_function_cie_file_interpolate (function), 950
cooling_function_cie_file_logtemperature_interpolate (function), 950
cooling_function_cie_file_read (subroutine), 950
cooling_function_cie_file_read_initialize (subroutine), 950
cooling_function_cmb_compton (subroutine), 951
cooling_function_cmb_compton_initialize (subroutine), 951
cooling_function_density_log_slope (function), 952
cooling_function_density_slope_atomic_cie_cloudy (subroutine), 954
cooling_function_density_slope_cie_file (subroutine), 950
cooling_function_density_slope_cmb_compton (subroutine), 952
cooling_function_density_slope_molecular_hydrogen_gp (subroutine), 955
cooling_function_gp_h2plus_electron (function), 954
cooling_function_gp_h_h2 (function), 955
cooling_function_gp_h_h2plus (function), 955
cooling_function_initialize (subroutine), 952
cooling_function_molecular_hydrogen_gp (subroutine), 955
cooling_function_molecular_hydrogen_gp_initialize (subroutine), 955
cooling_function_not_matched (subroutine), 953
cooling_function_temperature_log_slope (function), 953
cooling_function_temperature_slope_atomic_cie_cloudy (subroutine), 954
cooling_function_temperature_slope_cie_file (subroutine), 950
cooling_function_temperature_slope_cmb_compton (subroutine), 952
cooling_function_temperature_slope_molecular_hydrogen_gp (subroutine), 955
cooling_functions (module), 952
cooling_functions_atomic_cie_cloudy (module), 953
cooling_functions_cie_file (module), 949
cooling_functions_cmb_compton (module), 951
cooling_functions_molecular_hydrogen_galli_palla (module), 954
cooling_infall_radii (module), 967
cooling_radii (module), 956
cooling_radii_isothermal (module), 957
cooling_radii_simple (module), 958
cooling_radius (function), 956
cooling_radius_growth_rate (function), 956
cooling_radius_growth_rate_isothermal (function), 957
cooling_radius_growth_rate_simple (function), 958
cooling_radius_hot_halo_output (subroutine), 956
cooling_radius_hot_halo_output_count (subroutine), 956
cooling_radius_hot_halo_output_names (subroutine), 956
cooling_radius_initialize (subroutine), 956
cooling_radius_isothermal (function), 957
cooling_radius_isothermal_initialize (subroutine), 957
cooling_radius_isothermal_reset (subroutine), 957
cooling_radius_output_initialize (subroutine), 957
cooling_radius_root (function), 958
cooling_radius_simple (function), 958
cooling_radius_simple_initialize (subroutine), 958
cooling_radius_simple_reset (subroutine), 958
cooling_radius_solver_initialize (subroutine), 959
cooling_radius (function), 960
cooling_rate_cole2000 (function), 959
cooling_rate_cole2000_initialize (subroutine), 959
cooling_rate_hot_halo_output (subroutine), 960
cooling_rate_hot_halo_output_count (subroutine), 960
cooling_rate_hot_halo_output_names (subroutine), 960
cooling_rate_initialize (subroutine), 960
cooling_rate_modifier_cut_off (subroutine), 961
cooling_rate_output_initialize (subroutine), 961
cooling_rate_simple (function), 962
cooling_rate_simple_initialize (subroutine), 962
cooling_rate_simple_scaling (function), 962
cooling_rate_simple_scaling_initialize (subroutine), 962
cooling_rate_white_frenk (function), 961
cooling_rate_white_frenk_initialize (subroutine), 961
cooling_rate_zero (function), 963
cooling_rate_zero_initialize (subroutine), 963
cooling_rates (module), 959
cooling_rates_cole2000 (module), 959
cooling_rates_modifier_cut_off (module), 961
cooling_rates_simple (module), 961
cooling_rates_simple_scaling (module), 962
cooling_rates_white_frenk (module), 960
cooling_rates_zero (module), 962
cooling_specific_am_constant_rotation_initialize (subroutine), 970
cooling_specific_am_constant_rotation_reset (subroutine), 970
cooling_specific_am_mean_initialize (subroutine), 971
cooling_specific_angular_momenta (module), 969
cooling_specific_angular_momenta_constant_rotation (module), 970
cooling_specific_angular_momenta_mean (module), 970
cooling_specific_angular_momentum (function), 969
cooling_specific_angular_momentum_constant_rotation (function), 970
cooling_specific_angular_momentum_initialize (subroutine), 970
cooling_specific_angular_momentum_mean (function), 971
cooling_time (function), 963
cooling_time_available (function), 971
cooling_time_available_get_template (function), 971
cooling_time_available_get_template (interface), 971
cooling_time_available_halo_formation (function), 973
cooling_time_available_halo_formation_initialize (subroutine), 973
cooling_time_available_increase_rate (function), 971
cooling_time_available_increase_rate_halo_formation (function), 973
cooling_time_available_increase_rate_wf (function), 972
cooling_time_available_initialize (subroutine), 972
cooling_time_available_wf (function), 972
cooling_time_available_wf_initialize (subroutine), 972
cooling_time_available_white_frenk (module), 972
cooling_time_density_log_slope (function), 963
cooling_time_density_log_slope_simple (function), 964
cooling_time_initialize (subroutine), 963
cooling_time_simple (function), 964
cooling_time_simple_initialize (subroutine), 964
cooling_time_temperature_log_slope (function), 963
cooling_time_temperature_log_slope_simple (function), 964
cooling_times (module), 963
cooling_times_available (module), 971
cooling_times_available_halo_formation (module), 972
cooling_times_simple (module), 964
coordinate (type), 1335
cordinate_systems (module), 1154
cordinatecartesian (type), 1335
cordinatecylindrical (type), 1335
cordinates (module), 1334
cordinates_assign (subroutine), 1335
cordinates_assign_from (subroutine), 1335
cordinates_assign_to (subroutine), 1335
coordinates_cartesian_from_cartesian (subroutine), 1336
cordinates_cartesian_set_x (subroutine), 1336
coordinates_cartesian_set_y (subroutine), 1336
cordinates_cartesian_set_z (subroutine), 1336
coordinates_cartesian_to_cartesian (function), 1336
coordinates_cartesian_to_cylindrical (function), 1155
coordinates_cartesian_to_spherical (function), 1155
cordinates_cartesian_x (function), 1336
cordinates_cartesian_y (function), 1336
cordinates_cartesian_z (function), 1336
coordinates_cylindrical_from_cartesian (subroutine), 1336
coordinates_cylindrical_phi (function), 1337
cordinates_cylindrical_r (function), 1337
coordinates_cylindrical_set_phi (subroutine), 1337
coordinates_cylindrical_set_r (subroutine), 1337
coordinates_cylindrical_set_z (subroutine), 1337
coordinates_cylindrical_to_cartesian (function), 1337
coordinates_cylindrical_to_spherical (function), 1155
coordinates_cylindrical_z (function), 1337
coordinates_null_from (subroutine), 1337
coordinates_null_to (function), 1337
coordinates_spherical_from_cartesian (subroutine), 1338
coordinates_spherical_phi (function), 1338
coordinates_spherical_r (function), 1338
coordinates_spherical_set_phi (subroutine), 1338
coordinates_spherical_set_r (subroutine), 1338
coordinates_spherical_set_theta (subroutine), 1338
coordinates_spherical_theta (function), 1338
coordinates_spherical_to_cartesian (function), 1338
coordinates_spherical_to_cylindrical (function), 1155
cosinespherical (type), 1338
cosec (interface), 1229
cosecdouble (function), 1229
cosecdoublecomplex (function), 1229
cosine_integral (function), 1226
cosmological_mass_root_variance (function), 1868
cosmological_mass_root_variance_logarithmic_derivative (function), 1869
cosmological_mass_root_variance_plus_logarithmic_derivative (subroutine), 1869
cosmological_mass_variance_filtered_power_spectrum (module), 1842
cosmological_mass_variance_filtered_power_spectrum_initialize (subroutine), 1842
cosmological_mass_variance_filtered_power_spectrum_tabulate (subroutine), 1842
cosmological_mass_variance_fps_state_retrieve (subroutine), 1842
cosmological_mass_variance_fps_state_store (subroutine), 1842
cosmology.functions.F90 (file), 973
cosmology.functions.matter_dark_energy.F90 (file), 989
cosmology.functions.matter Lambda.F90 (file), 991
cosmology.functions.options.F90 (file), 996
cosmology.functions.parameters.F90 (file), 996
cosmology.parameters.F90 (file), 996
cosmology.parameters.simple.F90 (file), 1002
cosmology_conversion_factors (subroutine), 1116
cosmology_functions (module), 973
cosmology_functions_options (module), 996
cosmology_functions_parameters (module), 996
cosmology_parameters (module), 996
cosmologyfunctions (interface), 977
cosmologyfunctions_c (function), 977
cosmologyfunctionsclass (type), 977
cosmologyfunctionscomovingvolumeelementredshift (function), 977
cosmologyfunctionscomovingvolumeelementtime (function), 977
cosmologyfunctionsconstructordefault (function), 978
cosmologyfunctionsconstructornamed (function), 978
cosmologyfunctions:cosmictimeisvalidnull (function), 978
cosmologyfunctions:cosmictimenull (function), 978
cosmologyfunctions:density scaling:earlytimenull (subroutine), 978
cosmologyfunctions:destructor_c (subroutine), 978
cosmologyfunctions:distance angular null (function), 978
cosmologyfunctions:distance comoving convert null (function), 978
cosmologyfunctions:distance comoving null (function), 978
cosmologyfunctions:distance luminosity null (function), 979
cosmologyfunctions:domination epoch matter null (function), 979
cosmologyfunctions:do state retrieve (subroutine), 979
cosmologyfunctions:do state store (subroutine), 979
cosmologyfunctions:epoch time (function), 979
cosmologyfunctions:equation of state dark energy null (function), 979
cosmologyfunctions:expansion factor from redshift (function), 980
cosmologyfunctions:expansion factor is valid null (function), 980
cosmologyfunctions:expansion factor null (function), 980
cosmologyfunctions:expansion rate null (function), 980
cosmologyfunctions:exponent dark energy (function), 980
cosmologyfunctions:shuttle parameter epochal null (function), 980
cosmologyfunctions:shuttle parameter rate of change null (function), 980
cosmologyfunctions:initialize (subroutine), 980
cosmologyfunctions:is finalizable (function), 981
cosmologyfunctions:matter dark energy (interface), 981, 989
cosmologyfunctions:matter lambda (interface), 981, 991
cosmologyfunctions:someg dark energy epochal null (function), 981
cosmologyfunctions:someg matter epochal null (function), 981
cosmologyfunctions:someg matter rate of change null (function), 981
cosmologyfunctions:redshift from expansion factor (function), 981
cosmologyfunctions:staterestore (subroutine), 981
cosmologyfunctions:state store (subroutine), 982
cosmologyfunctions:temperature cmb epochal null (function), 982
cosmologyfunctions:time at distance comoving null (function), 982
cosmologyfunctions:wrapper (type), 982
cosmologyparameters: (interface), 999
cosmologyparameters:class (type), 999
cosmologyparameters:constructor default (function), 999
cosmologyparameters:constructor named (function), 999
cosmologyparameters:density critical null (function), 999
cosmologyparameters:shuttle constant null (function), 999
cosmologyparameters:initialize (subroutine), 1000
cosmologyparameters:is finalizable (function), 1000
cosmologyparameters:someg baryon null (function), 1000
cosmologyparameters:someg curv null (function), 1000
cosmologyparameters:someg dark energy null (function), 1000
cosmologyparameters:someg matter null (function), 1000
cosmologyparameters:someg radiation null (function), 1000
cosmologyparameters:simple (interface), 1000, 1002
cosmologyparameterssimpleconstructor (function), 1001, 1002
cosmologyparameterssimpledefaultconstructor (function), 1001, 1002
cosmologyparametersimpledestructor (subroutine), 1001, 1002
cosmologyparametersstemperaturecmbnull (function), 1001
cot (interface), 1229
cotdouble (function), 1229
cotdoublecomplex (function), 1230
count_lines_in_file (interface), 1935
count_lines_in_file_char (function), 1935
count_lines_in_file_varstr (function), 1935
count_properties (subroutine), 1121
create_branch_jump_event (subroutine), 1247
create_indentation_format (subroutine), 1090
create_node_array (subroutine), 1247
create_node_indices (subroutine), 1247
critical_overdensities_kitayama_suto1996 (module), 1845
critical_overdensity (module), 1843
critical_overdensity_collapsing_mass (function), 1844
critical_overdensity_for_collapse (function), 1844
critical_overdensity_for_collapse_time_gradient (function), 1844
critical_overdensity_initialize (subroutine), 1844
critical_overdensity_kitayama_suto1996 (subroutine), 1845
critical_overdensity_kitayama_suto1996_initialize (subroutine), 1845
critical_overdensity_mass_scaling (function), 1844
critical_overdensity_mass_scaling_gradient (function), 1844
critical_overdensity_mass_scaling_gradient_null (function), 1846
critical_overdensity_mass_scaling_gradient_wdm (function), 1846
critical_overdensity_mass_scaling_initialize (subroutine), 1845
critical_overdensity_mass_scaling_null (function), 1846
critical_overdensity_mass_scaling_null_initialize (subroutine), 1846
critical_overdensity_mass_scaling_wdm (function), 1848
critical_overdensity_mass_scaling_wdm_initialize (subroutine), 1848
critical_overdensity_mass_scalings_null (module), 1846
critical_overdensity_mass_scalings_wdm (module), 1846
critical_overdensity_state_retrieve (subroutine), 1845
cross_section_h2_gamma_to_2h (function), 939
cross_section_h2_gamma_to_h2plus_electron (function), 939
cross_section_h2plus_gamma_to_2hplus_electron (function), 939
cross_section_h2plus_gamma_to_h_hplus (function), 939
cross_section_h_gamma_to_hplus_electron (function), 939
cross_section_hminus_gamma_to_h_electron (function), 939
cross_section_integrand (function), 1723
cumulative_conditional_mass_function (function), 1161
cumulative_conditional_mass_function_behroozi2010 (function), 1160
cumulative_conditional_mass_function_compute (subroutine), 1160
cumulative_conditional_mass_function_var_behroozi2010 (function), 1161
cumulative_conditional_mass_function_variance (function), 1161
cumulative_energy_integrand (function), 1787
dark_matter_halo_angular_momentum (function), 1017
dark_matter_halo_angular_momentum_growth_rate (function), 1017
dark_matter_halo_bias (interface), 1860
dark_matter_halo_bias_by_mass (function), 1860
dark_matter_halo_bias_by_node (function), 1860
dark_matter_halo_bias_initialize (subroutine), 1860
dark_matter_halo_bias_node_press_schechter (function), 1861
dark_matter_halo_bias_node_smt (function), 1861
dark_matter_halo_bias_node_tinker2010 (function), 1862
dark_matter_halo_bias_press_schechter (function), 1861
dark_matter_halo_bias_press_schechter_initialize (subroutine), 1861
dark_matter_halo_bias_smt (function), 1861
dark_matter_halo_bias_smt_initialize (subroutine), 1862
dark_matter_halo_bias_tinker2010 (function), 1862
dark_matter_halo_bias_tinker2010_initialize (subroutine), 1862
dark_matter_halo_biases (module), 1860
dark_matter_halo_biases_press_schechter (module), 1860
dark_matter_halo_biases_smt (module), 1861
dark_matter_halo_biases_tinker2010 (module), 1862
dark_matter_halo_formation_times (module), 1003
dark_matter_halo_mass_accretion_histories (module), 1004
dark_matter_halo_mass_accretion_histories_wechsler2002 (module), 1004
dark_matter_halo_mass_accretion_histories_zhao2009 (module), 1005
dark_matter_halo_mass_accretion_time (function), 1004
dark_matter_halo_mass_accretion_time_wechsler2002 (function), 1005
dark_matter_halo_mass_accretion_time_zhao2009 (function), 1005
dark_matter_halo_mass_loss_rates_initialize (subroutine), 1006
dark_matter_halo_scales (module), 1007
dark_matter_halo_spins (module), 1017
dark_matter_halo_spins_initialize (subroutine), 1017
dark_matter_halos.formation_times.F90 (file), 1003
dark_matter_halos.mass_accretion_history.F90 (file), 1004
dark_matter_halos.mass_accretion_history.Wechsler2002.F90 (file), 1004
dark_matter_halos.mass_accretion_history.Zhao2009.F90 (file), 1005
dark_matter_halos.mass_loss_rates.F90 (file), 1006
dark_matter_halos.mass_loss_rates.null.F90 (file), 1006
dark_matter_halos.mass_loss_rates.vanDenBosch.F90 (file), 1007
dark_matter_halos.scales.F90 (file), 1007
dark_matter_halos.scales.virial_density_contrast.F90 (file), 1015
dark_matter_halos.spins.distributions.delta_function.F90 (file), 1019
dark_matter_halos.spins.distributions.F90 (file), 1018
dark_matter_halos.spins.distributions.lognormal.F90 (file), 1019
dark_matter_halos.spins.F90 (file), 1017
dark_matter_halos.mass_loss_rate (function), 1006
dark_matter_halos.mass_loss_rate_null (function), 1006
dark_matter_halos.mass_loss_rate_null_initialize (subroutine), 1007
dark_matter_halos.mass_loss_rate_vandenbosch (function), 1007
dark_matter_halos.mass_loss_rate_vandenbosch_initialize (subroutine), 1007
dark_matter_halos_mass_loss_rates (module), 1006
dark_matter_halos_mass_loss_rates_null (module), 1006
dark_matter_halos_mass_loss_rates_vandenbosch (module), 1007
dark_matter_mass_accretion_initialize (subroutine), 1004
dark_matter_mass_accretion_wechsler2002Initialize (subroutine), 1005
dark_matter_mass_accretion_zhao2009Initialize (subroutine), 1005
dark_matter_profile_density_task (function), 1065
dark_matter_profile_enclosed_mass_task (function), 1065
dark_matter_profile_potential_task (function), 1065
dark_matter_profile_rotation_curve_gradient_task (function), 1065
dark_matter_profile_rotation_curve_task (function), 1065
dark_matter_profile_scale (function), 1063
dark_matter_profile_scales (module), 1063
dark_matter_profile_shape (function), 1063
dark_matter_profile_shape_gao2008 (function), 1064
dark_matter_profile_structure_tasks (module), 1064
dark_matter_profiles (module), 1025
dark_matter_profiles.Einasto.F90 (file), 1020
dark_matter_profiles.error_codes.F90 (file), 1045
dark_matter_profiles.F90 (file), 1025
dark_matter_profiles.iso thermal.F90 (file), 1045
dark_matter_profiles.NFW.F90 (file), 1040
dark_matter_profiles.structure.concentration.F90 (file), 1050
dark_matter_profiles.structure.concentration.NFW.F90 (file), 1059
dark_matter_profiles.structure.concentration.Prada2011.F90 (file), 1060
dark_matter_profiles.structure.concentration.WDM.F90 (file), 1061
dark_matter_profiles.structure.concentration.Zhao2009.F90 (file), 1062
dark_matter_profiles.structure.scale.F90 (file), 1062
dark_matter_profiles.structure.shape.F90 (file), 1063
dark_matter_profiles.structure_tasks.F90 (file), 1064
dark_matter_profiles_concentration (module), 1050
dark_matter_profiles_error_codes (module), 1045
dark_matter_profiles_shapes (module), 1063
dark_matter_profiles_shapes_gao2008 (module), 1064
dark_matter_shapes_gao2008_initialize (subroutine), 1064
dark_matter_shapes_initialize (subroutine), 1064
darkmatterhaloscale (interface), 1011
darkmatterhaloscalecalculationreset (subroutine), 1011
darkmatterhaloscaleclass (type), 1011
darkmatterhaloscaleconstructordefault (function), 1011
darkmatterhaloscaleconstructornamed (function), 1011
darkmatterhaloscaledocalculationreset (subroutine), 1011
darkmatterhaloscaledostateretrieve (subroutine), 1011
darkmatterhaloscaledostatestore (subroutine), 1012
darkmatterhaloscaledynamicaltimescalenull (function), 1012
darkmatterhaloscaleinitialize (subroutine), 1012
darkmatterhaloscaleisfinalizable (function), 1012
darkmatterhaloscalemeandensitygrowthratenull (function), 1012
darkmatterhaloscalemeandensitynull (function), 1012
darkmatterhaloscalestatestore (subroutine), 1012
darkmatterhaloscalestaterestore (subroutine), 1012
darkmatterhaloscalevirialdensitycontrastdefinition (interface), 1012, 1015
darkmatterhaloscalevirialradiusgrowthratenull (function), 1013
darkmatterhaloscalevirialradiusnull (function), 1013
darkmatterhaloscalevirialtemperaturenull (function), 1013
darkmatterhaloscalevirialvelocitygrowthratenull (function), 1013
darkmatterhaloscalevirialvelocitynull (function), 1013
darkmatterprofile (interface), 1026
darkmatterprofilecalculationreset (subroutine), 1026
darkmatterprofilecircularvelocitymaximumnull (function), 1026
darkmatterprofilecircularvelocitynull (function), 1026
darkmatterprofileclass (type), 1026
darkmatterprofileconcentration (interface), 1050
darkmatterprofileconcentrationclass (type), 1050
darkmatterprofileconcentrationconcentrationnull (function), 1051
darkmatterprofileconcentrationconstructordefault (function), 1051
darkmatterprofileconcentrationconstructornamed (function), 1051
darkmatterprofileconcentrationdarkmatterprofiledefinitionnull (function), 1051
darkmatterprofileconcentrationdensitycontrastdefinitionnull (function), 1051
darkmatterprofileconcentrationdiemerkravtsov2014 (interface), 1048, 1051
darkmatterprofileconcentrationduttonmaccio2014 (interface), 1049, 1051
darkmatterprofileconcentrationgao2008 (interface), 1051, 1058
darkmatterprofileconcentrationinitialize (subroutine), 1051
darkmatterprofileconcentrationisfinalizable (function), 1052
darkmatterprofileconcentrationmunozcuartas2011 (interface), 1052, 1058
darkmatterprofileconcentrationnfw1996 (interface), 1052, 1059
darkmatterprofileconcentrationprada2011 (interface), 1052, 1060
darkmatterprofileconcentrationwdm (interface), 1052, 1061
darkmatterprofileconcentrationzhao2009 (interface), 1052, 1062
darkmatterprofileconstructordefault (function), 1026
darkmatterprofileconstructornamed (function), 1026
darkmatterprofilecountlinked (function), 1393
darkmatterprofilecreatebyinterrupt (subroutine), 1393
darkmatterprofilecreatelinked (subroutine), 1393
darkmatterprofiledensitynull (function), 1027
darkmatterprofiledestroylinked (subroutine), 1393
darkmatterprofilelocalcalculationreset (subroutine), 1027
darkmatterprofileostateretrieve (subroutine), 1027
darkmatterprofileostatestore (subroutine), 1027
darkmatterprofileeinasto (interface), 1020, 1027
darkmatterprofileenclosedmassnull (function), 1027
darkmatterprofileenergygrowthratenull (function), 1027
darkmatterprofileenergynull (function), 1027
diemerkravtsov2014constructor (function), 1048, 1052
diemerkravtsov2014darkmatterprofiledefinition (function), 1049, 1053
diemerkravtsov2014defaultconstructor (function), 1049, 1053
diemerkravtsov2014densitycontrastdefinition (function), 1049, 1053
diskabundancesgas (function), 1399
diskabundancesgasattributematch (function), 1399
diskabundancesgasiscgettable (function), 1399
diskabundancesgasrate (subroutine), 1399
diskabundancesstellar (function), 1399
diskabundancesstellarattributematch (function), 1399
diskabundancesstellariscgettable (function), 1400
diskangularmomentum (function), 1400
diskangularmomentumattributematch (function), 1400
diskangularmomentumiscgettable (function), 1400
diskangularmomentumrate (subroutine), 1400
diskcountlinked (function), 1400
diskcreatebyinterrupt (subroutine), 1400
diskcreatelinked (subroutine), 1400
diskdestroylinked (subroutine), 1401
diskexponentialabundancesgascount (function), 1401
diskexponentialabundancesgasget (function), 1401
diskexponentialabundancesgasrate (subroutine), 1401
diskexponentialabundancesgasrategeneric (subroutine), 1401
diskexponentialabundancesgasscale (subroutine), 1401
diskexponentialabundancesgasset (subroutine), 1401
diskexponentialabundancesstellarcount (function), 1401
diskexponentialabundancesstellarget (function), 1401
diskexponentialabundancesstellarrate (subroutine), 1402
diskexponentialabundancesstellarscale (subroutine), 1402
diskexponentialabundancesstellarset (subroutine), 1402
diskexponentialangularmomentumcount (function), 1402
diskexponentialangularmomentumget (function), 1402
diskexponentialangularmomentumrate (subroutine), 1402
diskexponentialangularmomentumrategeneric (subroutine), 1402
diskexponentialangularmomentumscale (subroutine), 1402
diskexponentialangularmomentumset (subroutine), 1403
diskexponentialluminositiesstellarcount (function), 1403
diskexponentialluminositiesstellarget (function), 1403
diskexponentialluminositiesstellarrate (subroutine), 1403
diskexponentialluminositiesstellarscale (subroutine), 1403
diskexponentialluminositiesstellarset (subroutine), 1403
diskexponentialmassgascount (function), 1404
diskexponentialmassgasget (function), 1404
diskexponentialmassgasrate (subroutine), 1404
diskexponentialmassgasrategeneric (subroutine), 1404
diskexponentialmassgasscale (subroutine), 1404
diskexponentialmassgasset (subroutine), 1404
CODE INDEX

diskexponentialmassstellarcount (function), 1404
diskexponentialmassstellarget (function), 1404
diskexponentialmassstellarrate (subroutine), 1404
diskexponentialmassstellarscale (subroutine), 1405
diskexponentialmassstellarset (subroutine), 1405
diskexponentialradiusget (function), 1405
diskexponentialradiusset (subroutine), 1405
diskexponentialstarformationhistorycount (function), 1405
diskexponentialstarformationhistoryget (function), 1405
diskexponentialstarformationhistoryrate (subroutine), 1405
diskexponentialstarformationhistoryscale (subroutine), 1405
diskexponentialstarformationhistoryset (subroutine), 1405
diskexponentialstarformationrateget (function), 1406
diskexponentialstarformationrategetfunction (subroutine), 1406
diskexponentialstarformationrategetisattached (function), 1406
diskexponentialstellarpropertieshistorycount (function), 1406
diskexponentialstellarpropertieshistoryget (function), 1406
diskexponentialstellarpropertieshistoryrate (subroutine), 1406
diskexponentialstellarpropertieshistoryscale (subroutine), 1406
diskexponentialstellarpropertieshistoryset (subroutine), 1406
diskexponentialstarformationrategetisattached (function), 1406
diskget (function), 1407
diskhalfmassradius (function), 1407
diskhalfmassradiusattributematch (function), 1407
diskhalfmassradiusisgettable (function), 1407
diskisinitialized (function), 1407
diskisinitializedattributematch (function), 1407
diskisinitializedisgettable (function), 1408
diskluminositiesstellar (function), 1408
diskluminositiesstellarattributematch (function), 1408
diskluminositiesstellarisgettable (function), 1408
diskmassgas (function), 1408
diskmassgasattributematch (function), 1408
diskmassgasisgettable (function), 1408
diskmassgasrate (subroutine), 1408
diskmassstellar (function), 1408
diskmassstellarattributematch (function), 1409
diskmassstellarisgettable (function), 1409
disknullbindingabundances0inout (function), 1409
disknullbindingdouble0inout (function), 1409
disknullbindinghistory0inout (function), 1409
disknullbindinginteger0in (function), 1409
disknullbindinglogical0inout (function), 1409
disknullbindingrateabundances0inout (subroutine), 1409
disknullbindingratedouble0inout (subroutine), 1409
disknullbindingratehistory0inout (subroutine), 1410
disknullbindingrateinteger0in (subroutine), 1410
disknullbindingratelogical0inout (subroutine), 1410
disknullbindingratesstellarluminosities0inout (subroutine), 1410
disknullbindingsetabundances0inout (subroutine), 1410
disknullbindingsetdouble0inout (subroutine), 1410
disknullbindingsethistory0inout (subroutine), 1410
disknullbindingsetinteger0inout (subroutine), 1410
disknullbindingsetstellarluminosities0inout (subroutine), 1410
disknullbindingstellarluminosities0inout (function), 1411
diskradius (function), 1411
diskradiusattributematch (function), 1411
diskradiusisgettable (function), 1411
diskstarformationhistory (function), 1411
diskstarformationhistoryattributematch (function), 1411
diskstarformationhistoryisgettable (function), 1411
diskstarformationrate (function), 1411
diskstarformationrateattributematch (function), 1411
diskstarformationrateisgettable (function), 1412
diskstellarpropertieshistory (function), 1412
diskstellarpropertieshistoryattributematch (function), 1412
diskstellarpropertieshistoryisgettable (function), 1412
diskvelocity (function), 1412
diskvelocityattributematch (function), 1412
diskvelocityisgettable (function), 1412
diskverysimplemassgascount (function), 1412
diskverysimplemassgasgasset (function), 1412
diskverysimplemassgasget (function), 1413
diskverysimplemassgasrate (subroutine), 1413
diskverysimplemassgasrategeneric (subroutine), 1413
diskverysimplemassgasscale (subroutine), 1413
diskverysimplemassgasset (subroutine), 1413
diskverysimplemassstellarcount (function), 1413
diskverysimplemassstellarrate (function), 1413
diskverysimplemassstellarscale (subroutine), 1413
diskverysimplemassstellarscale (subroutine), 1414
dlnrel (function), 1182
do_interpolation (function), 943, 951
double_buffer_dump (subroutine), 1122
double_buffer_extend (subroutine), 1122
dpdel (function), 1182
dpmpar (function), 947
dpni (subroutine), 1182
drcmp (function), 1182
drexp (function), 1182
drlog (function), 1182
dsin1 (function), 1182
dump_tree (subroutine), 1248
duttonmaccio2014concentration (function), 1049, 1053
duttonmaccio2014constructor (function), 1049, 1053
duttonmaccio2014darkmatterprofiledefinition (function), 1049, 1053
duttonmaccio2014defaultconstructor (function), 1050, 1053
duttonmaccio2014densitycontrastdefinition (function), 1050, 1053
dxprop (function), 947
dynamical_friction_acceleration_chandrasekhar (module), 1749
dynamical_friction_acceleration_null (module), 1750
dynamical_friction_timescale_multiplier (function), 1773
dynamical_friction_timescale_utilities (module), 1772
dynamicsstatisticsadiabaticratio (function), 1414
dynamicsstatisticsadiabaticratioattributematch (function), 1414
dynamicsstatisticsadiabaticratioisgettable (function), 1414
dynamicsstatisticsbarinstabilitytimescale (function), 1414
dynamicsstatisticsbarinstabilitytimescaleattributematch (function), 1414
dynamicsstatisticsbarinstabilitytimescaleisgettable (function), 1414
dynamicsstatisticsbarsadiabaticratioget (function), 1414
dynamicsstatisticsbarsadiabaticratioset (subroutine), 1415
dynamicsstatisticsbarsbarinstabilitytimescaleget (function), 1415
dynamicsstatisticsbarsbarinstabilitytimescaleset (subroutine), 1415
dynamicsstatisticsbarstimeset (subroutine), 1415
dynamicsstatisticscreatebyinterrupt (subroutine), 1415
dynamicsstatisticscreatelinked (subroutine), 1415
dynamicsstatisticsdestroylinked (subroutine), 1415
dynamicsstatisticsget (function), 1416
dynamicsstatisticsnullbindingdouble1inout (function), 1416
dynamicsstatisticsnullbindinginteger0in (function), 1416
dynamicsstatisticsnullbindingratedouble1inout (subroutine), 1416
dynamicsstatisticsnullbindingrateinteger0in (subroutine), 1416
dynamicsstatisticsnullbindingsetdouble1inout (subroutine), 1416
dynamicsstatisticsnullbindingsetinteger0in (subroutine), 1416
dynamicsstatisticsstatisticsetime (function), 1416
dynamicsstatisticsstatisticsetimeattributematch (function), 1417
dynamicsstatisticsstatisticsetimeisgettable (function), 1417
einastocalculationreset (subroutine), 1020, 1029
einastocircularvelocity (function), 1021, 1029
einastocircularvelocitymaximum (function), 1021, 1029
einastocircularvelocitypeakradius (function), 1021, 1029
einastocostructor (function), 1021, 1029
einastodefaultconstructor (function), 1021, 1030
einastodensity (function), 1021, 1030
einastodensityscalefree (function), 1021, 1030
einastodestructor (subroutine), 1021, 1030
einastoenclosedmass (function), 1022, 1030
einastoenclosedmassscalefree (function), 1022, 1030
einastoeenergy (function), 1022, 1030
einastoeenergygrowthrate (function), 1022, 1031
einastoeenergytablemake (subroutine), 1022, 1031
einastofourierprofileintegrand (function), 1023, 1031
einastofourierprofi1etablemake (subroutine), 1023, 1031
CODE INDEX

einstofreefallradius (function), 1023, 1032
einstofreefallradiusincreaserate (function), 1023, 1032
einstofreefalltabulate (subroutine), 1023, 1032
einstofreefalltimescalefree (function), 1023, 1032
einstofreefalltimescalefreeintegrand (function), 1024, 1032
einstojeansequationintegrand (function), 1022, 1031
einstokineticenergyintegrand (function), 1022, 1031
einstokspace (function), 1024, 1032
einstopotential (function), 1024, 1032
einstopotentialenergyintegrand (function), 1023, 1031
einstopotentialscalefree (function), 1024, 1033
einstoradiusfromspecificangularmomentum (function), 1024, 1033
einstoradiusfromspecificangularmomentumscalefree (function), 1024, 1033
einstoradiusfromspecificangularmomentumtablemake (subroutine), 1024, 1033
einstorotationnormalization (function), 1025, 1033
einstostaterestore (subroutine), 1025, 1033
einstostatetestore (subroutine), 1025, 1033
electron_density (function), 945
electron_density_atomic_cie_cloudy (function), 946
electron_density_cie_file (function), 943
electron_density_cie_file_interpolate (function), 944
electron_density_cie_file_logtemperature_interpolate (function), 944
electron_density_density_log_slope (function), 945
electron_density_density_log_slope_atomic_cie_cloudy (function), 946
electron_density_density_log_slope_cie_file (function), 944
electron_density_temperature_log_slope (function), 945
electron_density_temperature_log_slope_atomic_cie_cloudy (function), 946
electron_density_temperature_log_slope_cie_file (function), 944
closed_mass_root (function), 1072
enforce_subhalo_status (subroutine), 1248
epsln (function), 947
equivalent_circular_orbit_solver (function), 1776
erfapproximation (function), 1854
error_function (function), 1226
error_function_complementary (function), 1226
error_functions (module), 1226
establish_property_names (subroutine), 1122
event_halo_formation (subroutine), 1066
events.branch_jump.F90 (file), 1065
events.halo_formation.F90 (file), 1066
events.node_merger.single_level_hierarchy.F90 (file), 1066
events.node_promotion.index_shift.F90 (file), 1067
events.subhalo_promotion.F90 (file), 1067
events_halo_formation (module), 1066
events_node_merger (subroutine), 1276
events_node_merger_do_slh (subroutine), 1066
events_node_merger_initialize_slh (subroutine), 1066
events_node_mergers_slh (module), 1066
evolve_to_time (function), 1275
evolve_to_time_report (subroutine), 1281
Evolution_Sets.F90 (file), 882
excursion_sets_barrier (function), 1848
excursion_sets_barrier_critical_overdensity (function), 1849
excursion_sets_barrier_effective (function), 1854, 1858
excursion_sets_barrier_gradient (function), 1848
excursion_sets_barrier_gradient_critical_overdensity (function), 1849
excursion_sets_barrier_gradient_linear (function), 1850
excursion_sets_barrier_gradient_quadratic (function), 1850
excursion_sets_barrier_gradient_remap_null (subroutine), 1852
excursion_sets_barrier_gradient_remap_scale (subroutine), 1852
excursion_sets_barrier_gradient_remap_smt (subroutine), 1851
excursion_sets_barrier_initialize (subroutine), 1849
excursion_sets_barrier_linear (function), 1850
excursion_sets_barrier_name (function), 1849
excursion_sets_barrier_quadratic (function), 1850
excursion_sets_barrier_remap_null (subroutine), 1852
excursion_sets_barrier_remap_scale (subroutine), 1852
excursion_sets_barrier_remap_smt (subroutine), 1851
excursion_sets_barriers (module), 1848
excursion_sets_barriers_critical_overdensity (module), 1849
excursion_sets_barriers_critical_overdensity_initialize (subroutine), 1849
excursion_sets_barriers_linear (module), 1850
excursion_sets_barriers_linear_initialize (subroutine), 1850
excursion_sets_barriers_quadratic (module), 1850
excursion_sets_barriers_quadratic_initialize (subroutine), 1851
excursion_sets_barriers_remap_null (module), 1851
excursion_sets_barriers_remap_null_initialize (subroutine), 1852
excursion_sets_barriers_remap_scale (module), 1852
excursion_sets_barriers_remap_scale_initialize (subroutine), 1852
excursion_sets_barriers_remap_smt (module), 1851
excursion_sets_barriers_remap_smt_initialize (subroutine), 1851
excursion_sets_first_crossing_farahi (module), 1853
excursion_sets_first_crossing_farahi_initialize (subroutine), 1854
excursion_sets_first_crossing_farahi_read_file (subroutine), 1854
excursion_sets_first_crossing_farahi_write_file (subroutine), 1854
excursion_sets_first_crossing_linear_barrier (module), 1858
excursion_sets_first_crossing_linear_barrier_initialize (subroutine), 1858
excursion_sets_first_crossing_probability (function), 1853
excursion_sets_first_crossing_probability_farahi (function), 1854
excursion_sets_first_crossing_probability_linear (function), 1858
excursion_sets_first_crossing_probability_zhang_hui (function), 1855
excursion_sets_first_crossing_probability_zhang_hui_high (function), 1857
excursion_sets_first_crossing_rate (function), 1853
excursion_sets_first_crossing_rate_farahi (function), 1855
excursion_sets_first_crossing_rate_linear (function), 1858
excursion_sets_first_crossing_rate_tabulate_farahi (subroutine), 1855
excursion_sets_first_crossing_rate_zhang_hui (function), 1856
excursion_sets_first_crossing_rate_zhang_hui_high (function), 1857
excursion_sets_first_crossing_zhang_hui (module), 1855
excursion_sets_first_crossing_zhang_hui_high (module), 1857
excursion_sets_first_crossing_zhang_hui_high_initialize (subroutine), 1858
excursion_sets_first_crossing_zhang_hui_initialize (subroutine), 1856
excursion_sets_first_crossing_zhang_hui_utilities (module), 1856
excursion_sets_first_crossings (module), 1853
excursion_sets_first_crossings_initialize (subroutine), 1853
excursion_sets_maximum_sigma_test (subroutine), 1233
excursion_sets_non_crossing_rate (function), 1853
excursion_sets_non_crossing_rate_farahi (function), 1855
excursion_sets_non_crossing_rate_linear (function), 1860
excursion_sets_non_crossing_rate_zhang_hui (function), 1856
excursion_sets_non_crossing_rate_zhang_hui_high (function), 1858
exists_integer_scalar_ch (function), 1936
exists_integer_scalar_vs (function), 1936
expansion_factor_at_formation (function), 1005
expansionfactor_c (function), 982
expansionrateperturbation (function), 1877
exparg (function), 947
exponential_integrals (module), 1226
extract (interface), 1215
extract_ch (function), 1215
extract_vs (function), 1215
extremum_solver (function), 1776
factorial (function), 1227
factorials (module), 1227
fftw3 (module), 1307
fftw_wavenumber (function), 1307
fgabnd (function), 884
file_exists (interface), 1935
file_exists_char (function), 1935
file_exists_varstr (function), 1935
file_lock (function), 1936
file_unlock (subroutine), 1936
file_utilities (module), 1934
fileconstructor (function), 909, 910, 1184, 1191
filedefaultconstructor (function), 909, 910, 1184, 1192
filedestructor (subroutine), 909, 910
fileelectronfraction (function), 1184, 1192
fileloadfile (subroutine), 909, 910
fileneutralheliumfraction (function), 1185, 1192
fileneutralhydrogenfraction (function), 1185, 1192
filereaddata (subroutine), 1185, 1192
filesinglyionizedheliumfraction (function), 1185, 1192
filespectrum (function), 909, 911
filetemperature (function), 1185, 1192
filter_extent (function), 1183
filter_get_index (function), 1183
filter_lightcone_get_coordinates (function), 1127
filter_luminosity_integrand (function), 1828
filter_luminosity_integrand_ab (function), 1828
filter_name (function), 1183
filter_response (function), 1183
filter_response_load (subroutine), 1183
filter_vega_offset (function), 1184
filtertype (type), 1184
fixedconstructor (function), 1887, 1892
fixeddefaultconstructor (function), 1887, 1892
fixeddensitycontrast (function), 1887, 1892
fixeddensitycontrastrateofchange (function), 1887, 1892
fodeiv2 (module), 1289
fodeiv2_control (type), 1290
fodeiv2_control_free (subroutine), 1290
fodeiv2_control_hadjust (function), 1290
fodeiv2_control_init (function), 1290
fodeiv2_control_name (function), 1290
fodeiv2_control_scaled_new (function), 1290
fodeiv2_control_standard_new (function), 1290
fodeiv2_control_status (function), 1290
fodeiv2_control_y_new (function), 1290
fodeiv2_control_yp_new (function), 1290
fodeiv2_driver (type), 1290
fodeiv2_driver_alloc_scaled_new (function), 1290
fodeiv2_driver_alloc_standard_new (function), 1291
fodeiv2_driver_alloc_y_new (function), 1291
fodeiv2_driver_alloc_yp_new (function), 1291
fodeiv2_driver_apply (function), 1291
fodeiv2_driver_free (subroutine), 1291
fodeiv2_driver_reset (function), 1291
fodeiv2_driver_set_hmax (function), 1291
fodeiv2_driver_set_hmin (function), 1291
fodeiv2_driver_set_nmax (function), 1291
fodeiv2_driver_status (function), 1291
fodeiv2_evolve (type), 1291
fodeiv2_evolve_alloc (function), 1291
fodeiv2_evolve_apply (function), 1292
fodeiv2_evolve_free (subroutine), 1292
fodeiv2_evolve_reset (function), 1292
fodeiv2_evolve_status (function), 1292
fodeiv2_step (type), 1292
fodeiv2_step_alloc (function), 1292
fodeiv2_step_apply (function), 1292
fodeiv2_step_free (subroutine), 1292
fodeiv2_step_name (function), 1292
fodeiv2_step_order (function), 1292
fodeiv2_step_reset (function), 1292
fodeiv2_step_status (function), 1292
fodeiv2_step_type (type), 1293
fodeiv2_system (type), 1293
fodeiv2_system_free (subroutine), 1293
fodeiv2_system_init (function), 1293
fodeiv2_system_status (function), 1293
formationtimecole2000formationtime (function), 1417
formationtimecountlinked (function), 1417
formationtimecreatebyinterrupt (subroutine), 1417
formationtimecreatelinked (subroutine), 1417
formationtimedestroylinked (subroutine), 1417
formationtimeformationtime (function), 1417
formationtimeformationtimeattributematch (function), 1417
formationtimeformationtimeisgettable (function), 1418
formationtimemget (function), 1418
formationtimemnullbindingdouble0inout (function), 1418
formationtimemnullbindinginteger0in (function), 1418
formationtimemnullbindingratedouble0inout (subroutine), 1418
formationtimemnullbindinginteger0in (subroutine), 1418
formationtimemnullbindingsetdouble0inout (subroutine), 1418
formationtimemnullbindingsetinteger0in (subroutine), 1418
formatted_date_and_time (function), 1934
freefall_radii (module), 965
freefall_radii_dark_matter_halo (module), 965
freefall_radius (function), 965
freefall_radius_dark_matter_halo (function), 965
freefall_radius_dark_matter_halo_initialize (subroutine), 965
freefall_radius_growth_rate (function), 965
freefall_radius_growth_rate_dark_matter_halo (function), 966
freefall_radius_initialize (subroutine), 965
freefall_time_available_halo_formation (function), 967
freefall_time_available_halo_formation_initialize (subroutine), 967
freefall_time_available_increase_rate_halo_formation (function), 967
freefall_times_available_halo_formation (module), 967
friendsoffriendsconstructor (function), 1887, 1892
friendsoffriendsdefaultconstructor (function), 1887, 1893
friendsoffriendsdensitycontrast (function), 1887, 1893
friendsoffriendsdensityconstratrateofchange (function), 1887, 1893
fshmrinverse (function), 1161
g_1 (function), 1856
g_2 (function), 1856
g_2_integrand_zhang_hui (function), 1857
g_2_integrated (function), 1857
galactic_dynamics.bar_instability.ELN.F90 (file), 1067
galactic_dynamics.bar_instability.ELN_tidal.F90 (file), 1068
galactic_dynamics.bar_instability.F90 (file), 1068
galactic_dynamics.bar_instability.null.F90 (file), 1069
galactic_dynamics.bar_instabilities (module), 1069
galactic_dynamics.bar_instabilities.eln (module), 1067
galactic_dynamics.bar_instabilities.eln_initialize (subroutine), 1068
galactic_dynamics_bar_instabilities_eln_tidal (module), 1068
galactic_dynamics_bar_instabilities_eln_tidal_initialize (subroutine), 1068
galactic_dynamics_bar_instabilities_null (module), 1069
galactic_dynamics_bar_instabilities_null_initialize (subroutine), 1069
galactic_dynamics_bar_instability_initialize (subroutine), 1069
galactic_structure.density.F90 (file), 1070
galactic_structure.enclosed_mass.F90 (file), 1070
galactic_structure.options.F90 (file), 1072
galactic_structure.potential.F90 (file), 1075
galactic_structure.radius_solver.adiabatic.F90 (file), 1076
galactic_structure.radius_solver.F90 (file), 1075
galactic_structure.radius_solver.fixed.F90 (file), 1077
galactic_structure.radius_solver.initial_radii.adiabatic.F90 (file), 1078
galactic_structure.radius_solver.initial_radii.F90 (file), 1077
galactic_structure.radius_solver.initial_radii.static.F90 (file), 1080
galactic_structure.radius_solver.linear.F90 (file), 1081
galactic_structure.radius_solver.procedures.F90 (file), 1081
galactic_structure.radius_solver.simple.F90 (file), 1082
galactic_structure.rotation_curve.F90 (file), 1082
galactic_structure.rotation_curve.gradient.F90 (file), 1083
galactic_structure.surface_density.F90 (file), 1084
galactic_structure.velocity_dispersions.F90 (file), 1084
galactic_structure_component_type_decode (function), 1075
galactic_structure_densities (module), 1070
galactic_structure_density (function), 1070
galactic_structure_enclosed_mass (function), 1072
galactic_structure_enclosed_mass_defaults (subroutine), 1072
galactic_structure_enclosed_masses (module), 1071
galactic_structure_initial_radii (module), 1077
galactic_structure_initial_radii_adiaabatic (module), 1078
galactic_structure_initial_radii_adiaabatic_initialize (subroutine), 1079
galactic_structure_initial_radii_static (module), 1080
galactic_structure_initial_radii_static_initialize (subroutine), 1080
galactic_structure_mass_type_decode (function), 1075
galactic_structure_options (module), 1072
galactic_structure_potential (function), 1075
galactic_structure_potentials (module), 1075
galactic_structure_radii (module), 1075
galactic_structure_radii_adiaabatic (module), 1076
galactic_structure_radii_adiaabatic_initialize (subroutine), 1076
galactic_structure_radii_fixed (module), 1077
galactic_structure_radii_fixed_initialize (subroutine), 1077
galactic_structure_radii_initial_adiaabatic_compute_factors (subroutine), 1079
galactic_structure_radii_linear (module), 1081
galactic_structure_radii_linear_initialize (subroutine), 1081
galactic_structure_radii_simple (module), 1082
galactic_structure_radii_simple_initialize (subroutine), 1082
galactic_structure_radii_solve (subroutine), 1076
galactic_structure_radii_solve_adiaabatic (subroutine), 1076
galactic_structure_radii_solve_fixed (subroutine), 1077
galactic_structure_radii_solve_linear (subroutine), 1081
galactic_structure_radii_solve_simple (subroutine), 1082
galactic_structure_radius_enclosing_mass (function), 1072
galactic_structure_radius_initial (function), 1078
galactic_structure_radius_initial_adiabatic (function), 1079
galactic_structure_radius_initial_adiabatic_solver (function), 1079
galactic_structure_radius_initial_derivative (function), 1078
galactic_structure_radius_initial_derivative_adiabatic (function), 1080
galactic_structure_radius_initial_derivative_adiabatic_solver (function), 1080
galactic_structure_radius_initial_derivative_static (function), 1080
galactic_structure_radius_initial_initialize (subroutine), 1078
galactic_structure_radius_initial_static (function), 1080
galactic_structure_radius_solver_procedures (module), 1081
galactic_structure_rotation_curve (function), 1083
galactic_structure_rotation_curve_gradient (function), 1083
galactic_structure_rotation_curve_gradients (module), 1083
galactic_structure_rotation_curves (module), 1082
galactic_structure_surface_densities (module), 1084
galactic_structure_surface_density (function), 1084
galactic_structure_velocity_dispersion (function), 1085
galactic_structure_velocity_densities (module), 1084
galacticus (program), 882
galacticus.banner.F90 (file), 1085
galacticus.calculations_reset.F90 (file), 1085
galacticus.display.F90 (file), 1086
galacticus.error.F90 (file), 1092
Galacticus.F90 (file), 882
galacticus.input.path.F90 (file), 1111
galacticus.meta.compute_times.F90 (file), 1112
galacticus.meta.compute_times.file.F90 (file), 1113
galacticus.meta.evolver_profiler.F90 (file), 1113
galacticus.meta.tree_timing.F90 (file), 1114
galacticus.output.analysis.cosmology_scalings.F90 (file), 1116
galacticus.output.analysis.mass_dependent_size_distribution.F90 (file), 1116
galacticus.output.analysis.mass_functions.F90 (file), 1118
galacticus.output.build.F90 (file), 1120
galacticus.output.HDF5.F90 (file), 1115
galacticus.output.HDF5.open.F90 (file), 1115
galacticus.output.merger_tree.density_contrasts.F90 (file), 1124
galacticus.output.merger_tree.descendants.F90 (file), 1125
galacticus.output.merger_tree.F90 (file), 1121
galacticus.output.merger_tree.filters.F90 (file), 1126
galacticus.output.merger_tree.filters.lightcone.F90 (file), 1127
galacticus.output.merger_tree.filters.luminosities.F90 (file), 1128
galacticus.output.merger_tree.filters.stellar_mass.F90 (file), 1129
galacticus.output.merger_tree.filters.stellar_mass_morphology.F90 (file), 1129
galacticus.output.merger_tree.final_descendent.F90 (file), 1130
galacticus.output.merger_tree.half_light_properties.F90 (file), 1130
galacticus.output.merger_tree.half_mass_radius.F90 (file), 1131
galacticus.output.merger_tree.halo_model.F90 (file), 1132
galacticus.output.merger_tree.links.F90 (file), 1133
galacticus.output.merger_tree.main_branch.F90 (file), 1134
galacticus.output.merger_tree.mass_profile.F90 (file), 1135
galacticus.output.merger_tree.most_massive_progenitor.F90 (file), 1136
galacticus.output.merger_tree.redshifts.F90 (file), 1136
galacticus.output.merger_tree.rotation_curve.F90 (file), 1137
galacticus.output.merger_tree.satellite_extrema.F90 (file), 1138
galacticus.output.merger_tree.satellite_host.F90 (file), 1139
galacticus.output.merger_tree.satellite_status.F90 (file), 1140
galacticus.output.merger_tree.star_formation.F90 (file), 1141
galacticus.output.merger_tree.star_formation.metallicity_split.F90 (file), 1142
galacticus.output.merger_tree.star_formation.null.F90 (file), 1143
galacticus.output.merger_tree.tree_indices.F90 (file), 1144
galacticus.output.merger_tree.velocity_dispersion.F90 (file), 1144
galacticus.output.merger_tree.virial.F90 (file), 1148
galacticus.output.times.F90 (file), 1149
galacticus.output.version.F90 (file), 1151
galacticus.state.F90 (file), 1151
galacticus.tasks.basic.F90 (file), 1153
galacticus.tasks.evolve_tree.F90 (file), 1154
galacticus.tasks.F90 (file), 1153
galacticus_banner (module), 1085
galacticus_banner_show (subroutine), 1085
galacticus_build (module), 1120
galacticus_build_output (subroutine), 1120
galacticus_build_string (function), 1121
galacticus_calculations_reset (subroutine), 1085
galacticus_calculations_resets (module), 1085
galacticus_component_list (function), 1110
galacticus_display (module), 1086
galacticus_display_counter (subroutine), 1090
galacticus_display_counter_clear (subroutine), 1091
galacticus_display_counter_clear_lockless (subroutine), 1091
galacticus_display_counter_lockless (subroutine), 1091
galacticus_display_indent (interface), 1091
galacticus_display_indent_char (subroutine), 1091
galacticus_display_indent_varstr (subroutine), 1091
galacticus_display_message (interface), 1091
galacticus_display_message_char (subroutine), 1091
galacticus_display_message_varstr (subroutine), 1091
galacticus_display_unindent (subroutine), 1091
galacticus_error (module), 1092
galacticus_error_handler_register (subroutine), 1110
galacticus_error_report (interface), 1110
galacticus_error_report_char (subroutine), 1111
galacticus_error_report_varstr (subroutine), 1111
galacticus_extra_output_halo_fourier_profile (subroutine), 1132
<table>
<thead>
<tr>
<th>Function/Module</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>galacticus_growth_factor_output (subroutine)</td>
<td>1133</td>
</tr>
<tr>
<td>galacticus_gsl_error_handler (subroutine)</td>
<td>1111</td>
</tr>
<tr>
<td>galacticus_hdf5 (module)</td>
<td>1115</td>
</tr>
<tr>
<td>galacticus_input_path (function)</td>
<td>1112</td>
</tr>
<tr>
<td>galacticus_input_paths (module)</td>
<td>1111</td>
</tr>
<tr>
<td>galacticus_linear_power_spectrum_output (subroutine)</td>
<td>1133</td>
</tr>
<tr>
<td>galacticus_merger_tree_lightcone_geometry_initialize (subroutine)</td>
<td>1127</td>
</tr>
<tr>
<td>galacticus_merger_tree_output (subroutine)</td>
<td>1123</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_filter (function)</td>
<td>1126</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_filter_initialize (subroutine)</td>
<td>1126</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_filter_lightcone (subroutine)</td>
<td>1127</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_filter_lightcone_initialize (subroutine)</td>
<td>1128</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_filter_lightcones (module)</td>
<td>1127</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_filter_luminosities (module)</td>
<td>1128</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_filter_luminosity (subroutine)</td>
<td>1128</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_filter_luminosity_initialize (subroutine)</td>
<td>1129</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_filter_stellar_mass (subroutine)</td>
<td>1129</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_filter_stellar_mass_initialize (subroutine)</td>
<td>1129</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_filter_stellar_masses (module)</td>
<td>1129</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_filters (module)</td>
<td>1126</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_finalize (subroutine)</td>
<td>1123</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_make_group (subroutine)</td>
<td>1123</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_stllr_mss_mrrphlgse (module)</td>
<td>1129</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_stllr_mss_mrrphlgse (module)</td>
<td>1129</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_stllr_mss_mrrphlgse (module)</td>
<td>1129</td>
</tr>
<tr>
<td>galacticus_merger_tree_output_stllr_mss_mrrphlgse (module)</td>
<td>1130</td>
</tr>
<tr>
<td>galacticus_meta_compute_times (module)</td>
<td>1112</td>
</tr>
<tr>
<td>galacticus_meta_compute_times_file (module)</td>
<td>1113</td>
</tr>
<tr>
<td>galacticus_meta_evolver_profile (subroutine)</td>
<td>1114</td>
</tr>
<tr>
<td>galacticus_meta_evolver_profile_output (subroutine)</td>
<td>1114</td>
</tr>
<tr>
<td>galacticus_meta_tree_timing (module)</td>
<td>1113</td>
</tr>
<tr>
<td>galacticus_next_output_time (function)</td>
<td>1150</td>
</tr>
<tr>
<td>galacticus_nodes (module)</td>
<td>1362</td>
</tr>
<tr>
<td>galacticus_nodes_finalize (subroutine)</td>
<td>1418</td>
</tr>
<tr>
<td>galacticus_nodes_initialize (subroutine)</td>
<td>1419</td>
</tr>
<tr>
<td>galacticus_nodes_unique_id_set (subroutine)</td>
<td>1419</td>
</tr>
<tr>
<td>galacticus_output_analyses_cosmology_scalings (module)</td>
<td>1116</td>
</tr>
<tr>
<td>galacticus_output_analyses_mass_dpndnt_sz_dstrbtils (module)</td>
<td>1116</td>
</tr>
<tr>
<td>galacticus_output_analyses_mass_functions (module)</td>
<td>1119</td>
</tr>
<tr>
<td>galacticus_output_analyses_mass_dpndnt_sz_dstrbtils (subroutine)</td>
<td>1118</td>
</tr>
<tr>
<td>galacticus_output_analyses_mass_dpndnt_sz_dstrbtils_output (subroutine)</td>
<td>1118</td>
</tr>
<tr>
<td>galacticus_output_analyses_mass_functions (subroutine)</td>
<td>1119</td>
</tr>
<tr>
<td>galacticus_output_analyses_mass_functions_output (subroutine)</td>
<td>1119</td>
</tr>
<tr>
<td>galacticus_output_close_file (subroutine)</td>
<td>1116</td>
</tr>
<tr>
<td>galacticus_output_halo_model (subroutine)</td>
<td>1133</td>
</tr>
<tr>
<td>galacticus_output_halo_model_initialize (subroutine)</td>
<td>1133</td>
</tr>
<tr>
<td>galacticus_output_halo_model_names (subroutine)</td>
<td>1133</td>
</tr>
<tr>
<td>galacticus_output_halo_model_property_count (subroutine)</td>
<td>1133</td>
</tr>
<tr>
<td>galacticus_output_halo_models (module)</td>
<td>1132</td>
</tr>
</tbody>
</table>
galacticus_output_merger_tree (module), 1121
galacticus_output_most_massive_progenitor (subroutine), 1136
galacticus_output_most_massive_progenitor_initialize (subroutine), 1136
galacticus_output_most_massive_progenitor_names (subroutine), 1136
galacticus_output_most_massive_progenitor_property_count (subroutine), 1136
galacticus_output_most_massive_progenitors (module), 1136
galacticus_output_open (module), 1115
galacticus_output_open_file (subroutine), 1116
galacticus_output_redshift (function), 1150
galacticus_output_redshifts (subroutine), 1137
galacticus_output_redshifts_inititalize (subroutine), 1137
galacticus_output_redshifts_names (subroutine), 1137
egalacticus_output_redshifts_property_count (subroutine), 1137
galacticus_output_star_formation_histories (module), 1141
galacticus_output_star_formation_histories_initialze (subroutine), 1141
galacticus_output_time (function), 1150
galacticus_output_time_count (function), 1150
galacticus_output_time_index (function), 1150
galacticus_output_times (module), 1149
galacticus_output_tree_density_contrast (subroutine), 1124
galacticus_output_tree_density_contrast_inititalize (subroutine), 1125
galacticus_output_tree_density_contrast_names (subroutine), 1125
galacticus_output_tree_density_contrast_property_count (subroutine), 1125
galacticus_output_tree_descendents (subroutine), 1125
galacticus_output_tree_descendents_inititalize (subroutine), 1126
galacticus_output_tree_descendents_names (subroutine), 1126
galacticus_output_tree_descendents_property_count (subroutine), 1126
galacticus_output_tree_final_descendents (subroutine), 1130
galacticus_output_tree_final_descendents_inititalize (subroutine), 1130
galacticus_output_tree_final_descendents_names (subroutine), 1130
galacticus_output_tree_final_descendents_property_count (subroutine), 1130
galacticus_output_tree_half_light (subroutine), 1131
galacticus_output_tree_half_light_inititalize (subroutine), 1131
galacticus_output_tree_half_light_names (subroutine), 1131
galacticus_output_tree_half_light_properties (module), 1131
galacticus_output_tree_half_light_property_count (subroutine), 1131
galacticus_output_tree_half_mass (subroutine), 1132
galacticus_output_tree_half_mass_inititalize (subroutine), 1132
galacticus_output_tree_half_mass_names (subroutine), 1132
galacticus_output_tree_half_mass_property_count (subroutine), 1132
galacticus_output_tree_half_mass_radii (module), 1131
galacticus_output_tree_indices_inititalize (subroutine), 1144
galacticus_output_tree_lightcone (subroutine), 1128
galacticus_output_tree_lightcone_names (subroutine), 1128
galacticus_output_tree_lightcone_property_count (subroutine), 1128
galacticus_output_tree_links (subroutine), 1134
galacticus_output_tree_links_names (subroutine), 1134
galacticus_output_tree_links_property_count (subroutine), 1134
galacticus_output_tree_main_branch (subroutine), 1134
galacticus_output_tree_main_branch_initialize (subroutine), 1134
galacticus_output_tree_main_branch_names (subroutine), 1135
galacticus_output_tree_main_branch_property_count (subroutine), 1135
galacticus_output_tree_mass_profile (subroutine), 1135
galacticus_output_tree_mass_profile_initialize (subroutine), 1135
galacticus_output_tree_mass_profile_names (subroutine), 1135
galacticus_output_tree_mass_profile_property_count (subroutine), 1135
galacticus_output_tree_mass_profiles (module), 1135
galacticus_output_tree_rotation_curve (subroutine), 1137
galacticus_output_tree_rotation_curve_initialize (subroutine), 1138
galacticus_output_tree_rotation_curve_names (subroutine), 1138
galacticus_output_tree_rotation_curve_property_count (subroutine), 1138
galacticus_output_tree_satellite_extremum (subroutine), 1138
galacticus_output_tree_satellite_extremum_initialize (subroutine), 1138
galacticus_output_tree_satellite_extremum_names (subroutine), 1139
galacticus_output_tree_satellite_extremum_property_count (subroutine), 1139
galacticus_output_tree_satellite_host (subroutine), 1139
galacticus_output_tree_satellite_host_initialize (subroutine), 1139
galacticus_output_tree_satellite_host_names (subroutine), 1139
galacticus_output_tree_satellite_host_property_count (subroutine), 1140
galacticus_output_tree_satellite_status (subroutine), 1140
galacticus_output_tree_satellite_status_initialize (subroutine), 1140
galacticus_output_tree_satellite_status_names (subroutine), 1140
galacticus_output_tree_satellite_status_property_count (subroutine), 1140
galacticus_output_tree_tree_indices (subroutine), 1144
galacticus_output_tree_tree_indices_names (subroutine), 1144
galacticus_output_tree_tree_indices_property_count (subroutine), 1144
galacticus_output_tree_velocity_dispersion (subroutine), 1145
galacticus_output_tree_velocity_dispersion_initialize (subroutine), 1145
galacticus_output_tree_velocity_dispersion_names (subroutine), 1145
galacticus_output_tree_velocity_dispersion_property_count (subroutine), 1145
galacticus_output_tree_virial (subroutine), 1149
galacticus_output_tree_virial_initialize (subroutine), 1149
galacticus_output_tree_virial_names (subroutine), 1149
galacticus_output_tree_virial_property_count (subroutine), 1149
galacticus_output_trees_density_contrasts (module), 1124
galacticus_output_trees_descendents (module), 1125
galacticus_output_trees_final_descendents (module), 1130
galacticus_output_trees_line_of_sight_velocity_dispersion (function), 1145
galacticus_output_trees_links (module), 1134
galacticus_output_trees_main_branch (module), 1134
galacticus_output_trees_redshifts (module), 1136
galacticus_output_trees_rotation_curve (module), 1137
galacticus_output_trees_satellite_extremum (module), 1138
galacticus_output_trees_satellite_host (module), 1139
galacticus_output_trees_satellite_status (module), 1140
galacticus_output_trees_tree_indices (module), 1144
galacticus_output_trees_velocity_dispersion (module), 1145
galacticus_output_trees_velocity_dispersion_density_integrand (function), 1146
galacticus_output_trees_virial (module), 1149
galacticus_output_trees_vlcty_dispersion_vlcty_dnsty_intgrnd (function), 1146
galacticus_output_trees_vlcty_dsprsn_dnsty_srfc_intgrnd (function), 1146
galacticus_output_trees_vlcty_dsprsn_lambdar_intgrnd1 (function), 1146
galacticus_output_trees_vlcty_dsprsn_lambdar_intgrnd2 (function), 1148
galacticus_output_trees_vlcty_dsprsn_vlcty_dnsty_srfc_intgrnd (function), 1148
galacticus_previous_output_time (function), 1151
galacticus_signal_handler_sigfpe (subroutine), 1111
galacticus_signal_handler_sigint (subroutine), 1111
galacticus_signal_handler_sigsegv (subroutine), 1111
galacticus_state (module), 1151
galacticus_state_retrieve (subroutine), 1152
galacticus_state_snapshot (subroutine), 1152
galacticus_state_store (subroutine), 1152
galacticus_task_do (subroutine), 1153
galacticus_task_end (function), 1153
galacticus_task_evolve_tree (function), 1154
galacticus_task_start (function), 1153
galacticus_tasks (module), 1153
galacticus_tasks_basic (module), 1153
galacticus_tasks_evolve_tree (module), 1154
galacticus_time_per_tree (function), 1113
galacticus_time_per_tree_file (function), 1113
galacticus_time_per_tree_file_initialize (subroutine), 1113
galacticus_time_per_tree_initialize (subroutine), 1113
galacticus_verbosity_level (function), 1092
galacticus_verbosity_level_set (subroutine), 1092
galacticus_version (function), 1151
galacticus_version_output (subroutine), 1151
galacticus_versioning (module), 1151
galacticusangularmomenta3davailable (function), 1251, 1268
galacticusangularmomentaavailable (function), 1251, 1268
galacticusangularmomentaincludesubhalos (function), 1251, 1268
galacticusclose (subroutine), 1251, 1268
galacticuscubelength (function), 1251, 1268
galacticusdefaultconstructor (function), 1251, 1268
galacticusdestructor (subroutine), 1252, 1269
galacticusimport (subroutine), 1252, 1269
galacticusmassesincludesubhalos (function), 1252, 1269
galacticusnodect (function), 1252, 1269
galacticusopen (subroutine), 1252, 1269
galacticusparticlecountavailable (function), 1252, 1269
galacticuspositionsareperiodic (function), 1252, 1269
galacticuspositionsavailable (function), 1252, 1270
galacticusscaleradiiavailable (function), 1253, 1270
galacticusspin3davailable (function), 1253, 1270
galacticusspinavailable (function), 1253, 1270
galacticussubhalotrace (subroutine), 1253, 1270
galacticussubhalotracecount (function), 1253, 1270
galacticustreecount (function), 1253, 1270
galacticustreeindex (function), 1253, 1270
galacticustreeindicesread (subroutine), 1253, 1270
galacticustreesareselfcontained (function), 1254, 1271
galacticustreeshavesubhalos (function), 1254, 1271
galacticustreeweight (function), 1254, 1271
galacticusvelocitiesincludehubbleflow (function), 1254, 1271
galacticusvelocitydispersionavailable (function), 1254, 1271
galacticusvelocitymaximumavailable (function), 1254, 1271
gaminv (subroutine), 1182
gamma_function (function), 1228
gamma_function_incomplete (function), 1228
gamma_function_incomplete_complementary (function), 1228
gamma_function_logarithmic (function), 1228
gamma_functions (module), 1227
gao2008concentration (function), 1054, 1058
gao2008darkmatterprofiledefinition (function), 1054, 1058
gao2008defaultconstructor (function), 1054, 1058
gao2008densitycontrastdefinition (function), 1054, 1058
gaunt_factor (function), 916
gaunt_factor_initialize (subroutine), 916
gaunt_factor_sutherland1998 (function), 917
gaunt_factor_sutherland1998_initialize (subroutine), 917
gaunt_factors_sutherland1998 (module), 916
gaussian_distribution (function), 1225
gaussian_random (module), 1315
gaussian_random_free (subroutine), 1315
gaussian_random_get (function), 1315
generalized_press_schechter_branch_mass (function), 1233
generalized_press_schechter_branch_mass_root (function), 1233
generalized_press_schechter_branching (module), 1232
generalized_press_schechter_branching_initialize (subroutine), 1233
generalized_press_schechter_branching_maximum_step (function), 1233
generalized_press_schechter_branching_probability (function), 1233
generalized_press_schechter_subresolution_fraction (function), 1234
genericnullbindingabundances0inout (function), 1419
genericnullbindingdouble0inout (function), 1419
genericnullbindingrateabundances0inout (subroutine), 1419
genericnullbindingratedouble0inout (subroutine), 1419
genericnullbindingsetabundances0inout (subroutine), 1419
genericnullbindingsetdouble0inout (subroutine), 1419
gometry.coordinate_systems.F90 (file), 1154
gometry.surveys.Caputi-2011-UKIDSS-UDS.F90 (file), 1155
gometry.surveys.F90 (file), 1156
gometry.surveys.Li-White-2009-SDSS.F90 (file), 1157
gometry.surveys.Martin-2010-ALFALFA.F90 (file), 1158
gometry_survey_distance_maximum (function), 1156
gometry_survey_distance_maximum_caputi_2011_ukidss_uds (function), 1155
gometry_survey_distance_maximum_li_white_2009_sdss (function), 1157
geometry_survey_distance_maximum_martin_2010_alfalfa (function), 1158
geometry_survey_solid_angle (function), 1157
geometry_survey_solid_angle_caputi_2011_ukidss_uds (function), 1155
geometry_survey_solid_angle_li_white_2009_sdss (function), 1157
geometry_survey_solid_angle_martin_2010_alfalfa (function), 1158
geometry_survey_volume_maximum (function), 1157
geometry_survey_volume_maximum_caputi_2011_ukidss_uds (function), 1156
geometry_survey_volume_maximum_li_white_2009_sdss (function), 1158
geometry_survey_volume_maximum_martin_2010_alfalfa (function), 1158
geometry_survey_window_functions (subroutine), 1157
geometry_survey_window_functions_caputi_2011_ukidss_uds (subroutine), 1156
geometry_survey_window_functions_li_white_2009_sdss (subroutine), 1158
geometry_survey_window_functions_martin_2010_alfalfa (subroutine), 1159
geometry_surveys (module), 1156
geometry_surveys_caputi_2011_ukidss_uds (module), 1155
geometry_surveys_caputi_2011_ukidss_uds_initialize (subroutine), 1156
geometry_surveys_initialize (subroutine), 1157
geometry_surveys_li_white_2009_sdss (module), 1157
geometry_surveys_li_white_2009_sdss_initialize (subroutine), 1158
geometry_surveys_martin_2010_alfalfa (module), 1158
geometry_surveys_martin_2010_alfalfa_initialize (subroutine), 1159
get (interface), 1215
get_ (subroutine), 1215
get_argument (interface), 1933
get_argument_character (subroutine), 1933
get_argument_double (subroutine), 1933
get_argument_integer (subroutine), 1933
get_argument_logical (subroutine), 1933
get_argument_real (subroutine), 1933
get_argument_varying_string (subroutine), 1934
get_input_parameter (interface), 1948
get_input_parameter_array_size (function), 1949
get_input_parameter_char (subroutine), 1949
get_input_parameter_char_array (subroutine), 1949
get_input_parameter_double (subroutine), 1949
get_input_parameter_double_array (subroutine), 1949
get_input_parameter_double_c (subroutine), 1949
get_input_parameter_integer (subroutine), 1950
get_input_parameter_integer_array (subroutine), 1950
get_input_parameter_integer_c (subroutine), 1950
get_input_parameter_integer_long (subroutine), 1950
get_input_parameter_integer_long_array (subroutine), 1950
get_input_parameter_logical (subroutine), 1950
get_input_parameter_logical_array (subroutine), 1951
get_input_parameter_varstring (subroutine), 1951
get_interpolation (subroutine), 944, 951
get_new_assert_result (function), 1986
get_path (subroutine), 1112
get_set_ch (subroutine), 1215
get_set_vs (subroutine), 1215
get_temporary_string (subroutine), 1934
get_tree (subroutine), 1154
get_unit (subroutine), 1215
get_unit_set_ch (subroutine), 1215
get_unit_set_vs (subroutine), 1215
gratio (subroutine), 1182
growingconstructor (function), 1169, 1172
growingcorevirialdensityfunction (function), 1170, 1172
growingdefaultconstructor (function), 1170, 1173
growingdestructor (subroutine), 1170, 1173
growingradius (function), 1170, 1173
growingstaterestore (subroutine), 1170, 1173
growingstatestore (subroutine), 1170, 1173
growth_factor_simple_initialize (subroutine), 1867
growthrateodes (function), 1006
growthtableodes (function), 1867
h_electron_to_hminus_photon_rate_coefficient (function), 941
h_hminus_to_h2_electron_rate_coefficient (function), 941
half_mass_radius_root (function), 1248
half_mass_radius_root_cole2000 (function), 1756
halo_mass_fraction_integrand (function), 1863
halo_mass_fraction_integrated (function), 1863
halo_mass_function (module), 1862
halo_mass_function_close_file (subroutine), 1159
halo_mass_function_compute (subroutine), 1159
halo_mass_function_differential (function), 1863
halo_mass_function_differential_press_schechter (function), 1864
halo_mass_function_differential_tinker2008 (function), 1865
halo_mass_function_initialize (subroutine), 1863
halo_mass_function_integrand (function), 1863
halo_mass_function_integrated (function), 1863
halo_mass_function_modifier_simple_systematic (subroutine), 1865
halo_mass_function_modifiers_simple_systematic (module), 1865
halo_mass_function_open_file (subroutine), 1160
halo_mass_function_output (subroutine), 1160
halo_mass_function_press_schechter (module), 1863
halo_mass_function_press_schechter_initialize (subroutine), 1864
halo_mass_function_sheth_tormen (module), 1864
halo_mass_function_sheth_tormen_differential (function), 1864
halo_mass_function_sheth_tormen_initialize (subroutine), 1864
halo_mass_function_tasks (module), 1159
halo_mass_function_tinker2008 (module), 1864
halo_mass_function_tinker2008_initialize (subroutine), 1865
halo_mass_functions (program), 883
Halo_Mass_Functions.F90 (file), 883
halo_mass_functions.tasks.F90 (file), 1159
halo_model.conditional_mass_function.Behroozi2010.F90 (file), 1160
halo_model.conditional_mass_function.F90 (file), 1161
halo_occupancy_integrand (function), 1818
halo_occupancy_time_integrand (function), 1818
halo_spin_distribution_bett2007 (function), 1018
halo_spin_distribution_bett2007_initialize (subroutine), 1018
halo_spin_distribution_bett2007_snapshot (subroutine), 1018
halo_spin_distribution_bett2007_state_retrieve (subroutine), 1018
halo_spin_distribution_bett2007_state_store (subroutine), 1018
halo_spin_distribution_delta_function (function), 1019
halo_spin_distribution_delta_function_initialize (subroutine), 1019
halo_spin_distribution_lognormal (function), 1020
halo_spin_distribution_lognormal_initialize (subroutine), 1020
halo_spin_distribution_lognormal_snapshot (subroutine), 1020
halo_spin_distribution_lognormal_state_retrieve (subroutine), 1020
halo_spin_distribution_lognormal_state_store (subroutine), 1020
halo_spin_distribution_sample (function), 1019
halo_spin_distributions (module), 1018
halo_spin_distributions_bett2007 (module), 1018
halo_spin_distributions_delta_function (module), 1019
halo_spin_distributions_lognormal (module), 1019
halo_spin_sample_get_template (function), 1019
halo_spin_sample_get_template (interface), 1019
hash_md5 (function), 1938
hash_perfect_create (subroutine), 1938
hash_perfect_destroy (subroutine), 1938
hash_perfect_index (function), 1939
hash_perfect_is_present (function), 1939
hash_perfect_size (function), 1939
hash_perfect_value (function), 1939
hashes (module), 1936
hashes_cryptographic (module), 1938
hashes_perfect (module), 1938
hashperfect (type), 1939
hdf5object (type), 1917
histories (module), 1339
histories_state_retrieve (subroutine), 1339
histories_state_store (subroutine), 1339
history (type), 1340
history_add (function), 1340
history_builder (subroutine), 1340
history_clone (subroutine), 1340
history_combine (subroutine), 1340
history_create (subroutine), 1340
history_deserialize (subroutine), 1340
history_destroy (subroutine), 1340
history_divide (function), 1341
history_dump (subroutine), 1341
history_dump_raw (subroutine), 1341
history_exists (function), 1341

2096
history_extend (subroutine), 1341
history_increment (subroutine), 1341
history_is_zero (function), 1341
history_long_integer_builder (subroutine), 1342
history_long_integer_clone (subroutine), 1342
history_long_integer_create (subroutine), 1342
history_long_integer_destroy (subroutine), 1342
history_long_integer_dump (subroutine), 1342
history_long_integer_dump_raw (subroutine), 1342
history_long_integer_exists (function), 1342
history_long_integer_read_raw (subroutine), 1342
history_long_integer_reset (subroutine), 1343
history_long_integer_trim (subroutine), 1343
history_multiply (function), 1343
history_multiply_switched (function), 1343
history_read_raw (subroutine), 1343
history_reset (subroutine), 1343
history_serialize (subroutine), 1343
history_serialize_count (function), 1343
history_set_times (subroutine), 1344
history_set_to_unity (subroutine), 1344
history_subtract (function), 1344
history_timesteps (subroutine), 1344
history_trim (subroutine), 1344
hminus_electron_to_h2_electron_rate_coefficient (function), 941
hminus_hplus_to_2h_rate_coefficient (function), 941
hopkins2007constructor (function), 909, 910
hosthistorycountlinked (function), 1419
hosthistorycreatebyinterrupt (subroutine), 1420
hosthistorycreatelinked (subroutine), 1420
hosthistorydestroylinked (subroutine), 1420
hosthistoryhostmassmaximum (function), 1420
hosthistoryhostmassmaximumattributematch (function), 1420
hosthistoryhostmassmaximumisgettable (function), 1420
hosthistorynullbindingdouble0inout (function), 1420
hosthistorynullbindinginteger0in (function), 1420
hosthistorynullbindingratedouble0inout (function), 1420
hosthistorynullbindingsetinteger0in (function), 1420
hot_halo.cold_mode.density_profile.cored_isothermal.core_radius.F90 (file), 1161
hot_halo.cold_mode.density_profile.cored_isothermal.core_radius.virial_radius.F90 (file), 1162
hot_halo.mass_distribution.beta_profile.F90 (file), 1168
hot_halo.mass_distribution.cored.core_radius.F90 (file), 1169
hot_halo.mass_distribution.cored.core_radius.growing.F90 (file), 1172
hot_halo.mass_distribution.cored.core_radius.virial_radius_fraction.F90 (file), 1173
hot_halo.mass_distribution.F90 (file), 1163
hot_halo.mass_distribution.null.F90 (file), 1174
hot_halo.mass_distribution.Ricotti2000.F90 (file), 1167
hot_halo.ram_pressure_force.F90 (file), 1175
hot_halo.ram_pressure_force.Font2008.F90 (file), 1175
hot_halo.ram_pressure_force.null.F90 (file), 1176
hot_halo.ram_pressure_stripping.F90 (file), 1176
hot_halo.ram_pressure_stripping.Font2008.F90 (file), 1177
hot_halo.ram_pressure_stripping.timescale.F90 (file), 1177
hot_halo.ram_pressure_stripping.timescale.halo_dynamical_time.F90 (file), 1178
hot_halo.ram_pressure_stripping.timescale.ram_pressure_acceleration.F90 (file), 1178
hot_halo.ram_pressure_stripping.virial_radius.F90 (file), 1179
hot_halo.temperature_profile.F90 (file), 1180
hot_halo.temperature_profile.virial.F90 (file), 1180
hot_halo.cold_mode_density_ciso_corer_init (subroutine), 1162
hot_halo.cold_mode_density_ciso_corer_vf_initialize (subroutine), 1162
hot_halo.cold_mode_density_ciso_corer_vfrac (function), 1162
hot_halo.cold_mode_density_ciso_corer_virial_fraction (module), 1162
hot_halo.cold_mode_density_cored_isothermal_core_radii (module), 1162
hot_halo.cold_mode_density_cored_isothermal_core_radius (function), 1162
hot_halo.mass_distribution_density_task (function), 1164
hot_halo.mass_distribution_rotation_curve_gradient_task (function), 1165
hot_halo.mass_distribution_rotation_curve_task (function), 1165
hot_halo.mass_distributions (module), 1163
hot_halo.mass_distributions_core_radii (module), 1169
hot_halo.ram_pressure_force (function), 1175
hot_halo.ram_pressure_force.Font2008 (module), 1175
hot_halo.ram_pressure_force.Font2008.get (function), 1176
hot_halo.ram_pressure_force.Font2008.initialize (subroutine), 1176
hot_halo.ram_pressure_force.null (module), 1176
hot_halo.ram_pressure_force.null.get (function), 1176
hot_halo.ram_pressure_force.null.initialize (subroutine), 1176
hot_halo.ram_pressure_forces (module), 1175
hot_halo.ram_pressure_stripping (module), 1176
hot_halo.ram_pressure_stripping.Font2008 (module), 1177
hot_halo.ram_pressure_stripping.Font2008.get (function), 1177
hot_halo.ram_pressure_stripping.Font2008.initialize (subroutine), 1177
hot_halo.ram_pressure_stripping.radius (function), 1177
hot_halo.ram_pressure_stripping.radius_solver (function), 1177
hot_halo.ram_pressure_stripping.timescale (function), 1178
hot_halo.ram_pressure_stripping.timescales (module), 1178
hot_halo.ram_pressure_stripping.virial_radii (module), 1179
hot_halo.ram_pressure_stripping.virial_radii.initialize (subroutine), 1179
hot_halo.ram_pressure_stripping.virial_radius (function), 1179
hot_halo.ram_pressure_timescale.halo_dyntime (function), 1178
hot_halo.ram_pressure_timescales.halo_dyntime (module), 1178
hot_halo.ram_pressure_timescales.halo_dyntime.initialize (subroutine), 1178

2098
hot_halo_ram_pressure_timescales_ram_pressure_accel (module), 1178
hot_halo_ram_pressure_timescales_ram_pressure_accel_initialize (subroutine), 1179
hot_halo_temperature (function), 1180
hot_halo_temperature_get_template (function), 1180
hot_halo_temperature_get_template (interface), 1180
hot_halo_temperature_initialize (subroutine), 1180
hot_halo_temperature_logarithmic_slope (function), 1180
hot_halo_temperature_logarithmic_slope_virial_get (function), 1181
hot_halo_temperature_profile (module), 1180
hot_halo_temperature_profile_virial (module), 1180
hot_halo_temperature_virial (subroutine), 1181
hot_halo_temperature_virial_get (function), 1181
hot_mode_fraction (function), 1683
hothaloabundances (function), 1421
hothaloabundancesattributematch (function), 1421
hothaloabundancescold (function), 1421
hothaloabundancescoldattributematch (function), 1422
hothaloabundancescolddisgettable (function), 1422
hothaloabundancescoldrate (subroutine), 1422
hothaloabundancesiscgettable (function), 1422
hothaloabundancesrate (subroutine), 1422
hothaloangularmomentum (function), 1422
hothaloangularmomentumattributematch (function), 1422
hothaloangularmomentumcold (function), 1422
hothaloangularmomentumcoldattributematch (function), 1423
hothaloangularmomentumcolddisgettable (function), 1423
hothaloangularmomentumcoldrate (subroutine), 1423
hothaloangularmomentumisgettable (function), 1423
hothaloangularmomentumrate (subroutine), 1423
hothalochemicals (function), 1423
hothalochemicalsisgettable (function), 1423
hothalochemicalsrate (subroutine), 1424
hothalochemistrycoldmodecoldcount (function), 1424
hothalochemistrycoldmodecoldget (function), 1424
hothalochemistrycoldmodecoldrate (subroutine), 1424
hothalochemistrycoldmodecoldset (subroutine), 1424
hothalochemistrycoldmodeangularmomentumcoldcount (function), 1424
hothalochemistrycoldmodeangularmomentumcoldget (function), 1424
hothalochemistrycoldmodeangularmomentumcoldrate (subroutine), 1425
hothalochemistrycoldmodeangularmomentumcoldscale (subroutine), 1425
hothalochemistrycoldmodeangularmomentumcoldset (subroutine), 1425
hothalochemistrycoldmasscoldcount (function), 1425
hothalochemistrycoldmasscoldget (function), 1425
hothalochemistrycoldmasscoldrate (subroutine), 1425
hothalochemistrycoldmasscoldscale (subroutine), 1425
hothalochemistrycoldmasscoldset (subroutine), 1425
hothalochemistrycoldmodeouterradiusgrowthrate (function), 1425
hothalocoldmodeouterradiusgrowthratedeferredfunctionset (subroutine), 1426
hothalocoldmodeouterradiusgrowthratedfrdfncntisset (function), 1426
hothalocoldmodeoutflowreturn (subroutine), 1426
hothalocoldmodeoutflowreturndeferredfunctionset (subroutine), 1426
hothalocoldmodeoutflowreturndfrrdfncntisset (function), 1426
hothalocountlinked (function), 1426
hothalocreatebyinterrupt (subroutine), 1426
hothalocreatelinked (subroutine), 1426
hothalodestroylinked (subroutine), 1427
hothaloget (function), 1427
hothaloheatsource (function), 1427
hothaloheatsourceattributematch (function), 1427
hothaloheatsourceisgettable (function), 1427
hothalohothalocoolingabundances (function), 1427
hothalohothalocoolingabundancesattributematch (function), 1427
hothalohothalocoolingabundancesisgettable (function), 1427
hothalohothalocoolingabundancesratefunction (subroutine), 1427
hothalohothalocoolingabundancesrateisattached (function), 1428
hothalohothalocoolingangularmomentum (function), 1428
hothalohothalocoolingangularmomentumattributematch (function), 1428
hothalohothalocoolingangularmomentumisgettable (function), 1428
hothalohothalocoolingangularmomentumratefunction (subroutine), 1428
hothalohothalocoolingangularmomentumrateisattached (function), 1428
hothalohothalocoolingmass (function), 1428
hothalohothalocoolingmassattributematch (function), 1428
hothalohothalocoolingmassisgettable (function), 1429
hothalohothalocoolingmassratefunction (subroutine), 1429
hothalohothalocoolingmassrateisattached (function), 1429
hothaloisinitialized (function), 1429
hothaloisinitializedattributematch (function), 1429
hothaloisinitializedisgettable (function), 1429
hothalomass (function), 1429
hothalomassattributematch (function), 1429
hothalomasscold (function), 1430
hothalomasscoldattributematch (function), 1430
hothalomasscoldisgettable (function), 1430
hothalomasscoldrate (subroutine), 1430
hothalomassdistribution (interface), 1165
hothalomassdistributionbetaprofile (interface), 1165, 1169
hothalomassdistributionclass (type), 1165
hothalomassdistributionconstructordefault (function), 1165
hothalomassdistributionconstructornamed (function), 1165
hothalomassdistributioncoreradius (interface), 1170
hothalomassdistributioncoreradiusclass (type), 1170
hothalomassdistributioncoreradiusconstructordefault (function), 1170
hothalomassdistributioncoreradiusconstructornamed (function), 1171
hothalomassdistributioncoreradiussdostateretrieve (subroutine), 1171
hothalomassdistributioncoreradiussdostatestore (subroutine), 1171
hothalomassdistributioncoreradiusgrowing (interface), 1171, 1173
hothalomassdistributioncoreradiusinitialize (subroutine), 1171
hothalomassdistributioncoreradiusisfinalizable (function), 1171
hothalomassdistributioncoreradiussradiusnull (function), 1171
hothalomassdistributioncoreradiusstaterestore (subroutine), 1171
hothalomassdistributioncoreradiustatestore (subroutine), 1172
hothalomassdistributioncoreradiusvirialfraction (interface), 1172, 1173
hothalomassdistributiondensitylogslopenull (function), 1165
hothalomassdistributiondensitynull (function), 1165
hothalomassdistributionenclosedmassnull (function), 1166
hothalomassdistributioninitialize (subroutine), 1166
hothalomassdistributionisfinalizable (function), 1166
hothalomassdistributionnull (interface), 1166, 1174
hothalomassdistributionradialmomentnull (function), 1166
hothalomassdistributionricotti2000 (interface), 1166, 1167
hothalomassdistributionrotationnormalizationnull (function), 1166
hothalomasssisgettable (function), 1430
hothalomassrate (subroutine), 1430
hothalomasssink (function), 1430
hothalomasssinkattributematch (function), 1430
hothalomasssinkisgettable (function), 1430
hothalonnubbindingabundances0inout (function), 1431
hothalonnubbindingchemicalabundances0inout (function), 1431
hothalonnubbindingdouble0inout (function), 1431
hothalonnubbindinginteger0in (function), 1431
hothalonnubbindinglogical0inout (function), 1431
hothalonnubbindingrateabundances0inout (subroutine), 1431
hothalonnubbindingratechemicalabundances0inout (subroutine), 1431
hothalonnubbindingratedouble0inout (subroutine), 1431
hothalonnubbindingrateinteger0in (subroutine), 1431
hothalonnubbindingratelogical0inout (subroutine), 1432
hothalonnubbindingsetabundances0inout (subroutine), 1432
hothalonnubbindingsetchemicalabundances0inout (subroutine), 1432
hothalonnubbindingsetdouble0inout (subroutine), 1432
hothalonnubbindingsetinteger0in (subroutine), 1432
hothalonnubbindingsetlogical0inout (subroutine), 1432
hothalouterradius (function), 1432
hothalouterradiusattributematch (function), 1432
hothalouterradiusisgettable (function), 1432
hothaloulflowedabundances (function), 1432
hothaloulflowedabundancesattributematch (function), 1433
hothaloulflowedabundancesisgettable (function), 1433
hothaloulflowedangularmomentum (function), 1433
hothaloulflowedangularmomentumattributematch (function), 1433
hothaloulflowedangularmomentumisgettable (function), 1433
hothaloulflowedmass (function), 1433
hothaloulflowedmassattributematch (function), 1433
hothaloulflowedmassisgettable (function), 1433
hothaloulflowingabundances (function), 1434
hothaloulflowingabundancesattributematch (function), 1434
hothalooutflowingabundancesisgettable (function), 1434
hothalooutflowingangularmomentum (function), 1434
hothalooutflowingangularmomentumattributematch (function), 1434
hothalooutflowingangularmomentumisgettable (function), 1434
hothalooutflowingmass (function), 1434
hothalooutflowingmassattributematch (function), 1434
hothalooutflowingmassisgettable (function), 1435
hothalooutflowtrackingouterradiussgrowthrate (function), 1435
hothalooutflowtrackingouterradiussgrowthratedeferredfunctionset (subroutine), 1435
hothalooutflowtrackingouterradiussgrowthratedfrrdfnctnisset (function), 1435
hothalooutflowtrackingoutflowreturn (subroutine), 1435
hothalooutflowtrackingoutflowreturndeferredfunctionset (subroutine), 1435
hothalooutflowtrackingoutflowreturnfrrdfnctnisset (function), 1435
hothalooutflowtrackingtrackedoutflowabundancescount (function), 1435
hothalooutflowtrackingtrackedoutflowabundancesget (function), 1436
hothalooutflowtrackingtrackedoutflowabundancesrate (subroutine), 1436
hothalooutflowtrackingtrackedoutflowabundancesscale (subroutine), 1436
hothalooutflowtrackingtrackedoutflowabundancesset (subroutine), 1436
hothalooutflowtrackingtrackedoutflowmasscount (function), 1436
hothalooutflowtrackingtrackedoutflowmassget (function), 1436
hothalooutflowtrackingtrackedoutflowmassrate (subroutine), 1436
hothalooutflowtrackingtrackedoutflowmassscale (subroutine), 1436
hothalooutflowtrackingtrackedoutflowmassset (subroutine), 1437
hothalostandardabundancescount (function), 1437
hothalostandardabundancesget (function), 1437
hothalostandardabundancesrate (subroutine), 1437
hothalostandardabundancesscale (subroutine), 1437
hothalostandardabundancesset (subroutine), 1437
hothalostandardangularmomentumcount (function), 1437
hothalostandardangularmomentumget (function), 1437
hothalostandardangularmomentumrate (subroutine), 1437
hothalostandardangularmomentumratescale (subroutine), 1438
hothalostandardangularmomentumset (subroutine), 1438
hothalostandardchemicalscount (function), 1438
hothalostandardchemicalsget (function), 1438
hothalostandardchemicalsrateset (subroutine), 1438
hothalostandardchemicalsscale (subroutine), 1438
hothalostandardchemicalsset (subroutine), 1438
hothalostandardcreatefunctionsset (subroutine), 1438
hothalostandardheatsourcerate (subroutine), 1438
hothalostandardheatsourceratefunction (subroutine), 1439
hothalostandardheatsourcerateisattached (function), 1439
hothalostandardhothalocoolingabundancesrate (subroutine), 1439
hothalostandardhothalocoolingangularmomentumrate (subroutine), 1439
hothalostandardhothalocoolingmassrate (subroutine), 1439
hothalostandardinitializeget (function), 1439
hothalostandardinitializedset (subroutine), 1439
hothalostandardmasscount (function), 1439
hothalostandardmassget (function), 1439
hothalostandardmassrate (subroutine), 1440
hothalostandardmassscale (subroutine), 1440
hothalostandardmassset (subroutine), 1440
hothalostandardmasssinkrate (subroutine), 1440
hothalostandardmasssinkratefunction (subroutine), 1440
hothalostandardmasssinkrateisattached (function), 1440
hothalostandardouterradiuscount (function), 1440
hothalostandardouterradiusget (function), 1440
hothalostandardouterradiusgetfunction (subroutine), 1440
hothalostandardouterradiusgetisattached (function), 1441
hothalostandardouterradiusgetvalue (function), 1441
hothalostandardouterradiusgrowthrate (function), 1441
hothalostandardouterradiusgrowthratedeferredfunctionset (subroutine), 1441
hothalostandardouterradiusgrowthratedfrrdfnctnisset (function), 1441
hothalostandardouterradiusrate (subroutine), 1441
hothalostandardouterradiusrate (subroutine), 1441
hothalostandardouterradiusrate (subroutine), 1441
hothalostandardouterradiusrate (subroutine), 1441
hothalostandardouterradiusget (function), 1442
hothalostandardoutflowedabundancescount (function), 1442
hothalostandardoutflowedabundancesget (function), 1442
hothalostandardoutflowedabundancesrate (subroutine), 1442
hothalostandardoutflowedabundancesset (subroutine), 1442
hothalostandardoutflowedabundancesesscale (subroutine), 1442
hothalostandardoutflowedabundancescount (function), 1442
hothalostandardoutflowedabundancesget (function), 1442
hothalostandardoutflowedabundancesrate (subroutine), 1442
hothalostandardoutflowedabundancesset (subroutine), 1442
hothalostandardoutflowedabundancesesscale (subroutine), 1442
hothalostandardoutflowedabundancescount (function), 1442
hothalostandardoutflowedabundancesget (function), 1442
hothalostandardoutflowedabundancesrate (subroutine), 1442
hothalostandardoutflowedabundancesset (subroutine), 1442
hothalostandardoutflowedmasscount (function), 1443
hothalostandardoutflowedmassget (function), 1443
hothalostandardoutflowedmassrate (subroutine), 1443
hothalostandardoutflowedmassscale (subroutine), 1443
hothalostandardoutflowedmassset (subroutine), 1443
hothalostandardoutflowingabundancesrate (subroutine), 1443
hothalostandardoutflowingabundancesratefunction (subroutine), 1443
hothalostandardoutflowingabundancesrateisattached (function), 1444
hothalostandardoutflowingangularmomentumrate (subroutine), 1444
hothalostandardoutflowingangularmomentumratefunction (subroutine), 1444
hothalostandardoutflowingangularmomentumrateisattached (function), 1444
hothalostandardoutflowingmassrate (subroutine), 1444
hothalostandardoutflowingmassratefunction (subroutine), 1444
hothalostandardoutflowreturn (subroutine), 1444
hothalostandardoutflowreturndeferredfunctionset (subroutine), 1445
hothalostandardoutflowreturnfrrdfnctnisset (function), 1445
hothalostandardstrippedabundancescount (function), 1445
hothalostandardstrippedabundancesget (function), 1445
hothalostandardstrippedabundancesrate (subroutine), 1445
hothalostandardstrippedabundancesesscale (subroutine), 1445
hothalostandardstrippedabundancesset (subroutine), 1445
hothalostandardstrippedmasscount (function), 1445
hothalostandardstrippedmassget (function), 1445
hothalostandardstrippedmassrate (subroutine), 1446
hothalostandardstrippedmassscale (subroutine), 1446
hothalostandardstrippedmassset (subroutine), 1446
hothalostandardunaccretedmasscount (function), 1446
hothalostandardunaccretedmassget (function), 1446
hothalostandardunaccretedmassrate (subroutine), 1446
hothalostandardunaccretedmassscale (subroutine), 1446
hothalostandardunaccretedmassset (subroutine), 1446
hothalostripedabundances (function), 1447
hothalostripedabundancesattributematch (function), 1447
hothalostripedabundancesisgettable (function), 1447
hothalostripedmass (function), 1447
hothalostripedmassattributematch (function), 1447
hothalostripedmassisgettable (function), 1447
hothalotrackedorflowabundances (function), 1447
hothalotrackedorflowabundancesattributematch (function), 1447
hothalotrackedorflowabundancesisgettable (function), 1448
hothalotrackedorflowmass (function), 1448
hothalotrackedorflowmassattributematch (function), 1448
hothalotrackedorflowmassisgettable (function), 1448
hothalounaccretedmass (function), 1448
hothalounaccretedmassattributematch (function), 1448
hothalounaccretedmassisgettable (function), 1448
hothalounaccretedmassrate (subroutine), 1448
hothalounaccretedmassrate (subroutine), 1449
hothaloverysimplehothalocoolingmassrate (subroutine), 1449
hothaloverysimplemasscount (function), 1449
hothaloverysimplemassget (function), 1449
hothaloverysimplemassrate (subroutine), 1449
hothaloverysimplemassrate (subroutine), 1449
hothaloverysimplemassscale (subroutine), 1449
hothaloverysimplemassset (subroutine), 1449
hothaloverysimpleouterradiusget (function), 1449
hothaloverysimpleouterradiusgetfunction (subroutine), 1449
hothaloverysimpleouterradiusgetisattached (function), 1449
hothaloverysimpleoutflowingmassrate (subroutine), 1450
hothaloverysimpleoutflowingmassratefunction (subroutine), 1450
hothaloverysimpleoutflowingmassrateisattached (function), 1450
hubbleconstantsimple (function), 1001, 1003
hypergeometric_if1 (function), 1229
hypergeometric_2f1 (function), 1229
hypergeometric_functions (module), 1228

iachar (interface), 1216
iachar_ (function), 1216
ichar (interface), 1216
ichar_ (function), 1216
idbvip (subroutine), 923
idcldp (subroutine), 923
ideal_gas_jeans_length (function), 1912
ideal_gas_sound_speed (function), 1912
ideal_gases_thermodynamics (module), 1912
identityapply (subroutine), 1835, 1841
identitydefaultconstructor (function), 1836, 1841
idgrid (subroutine), 923
idlctn (subroutine), 923
idpdrv (subroutine), 923
idptip (subroutine), 923
idsfft (subroutine), 923
idtang (subroutine), 924
idxchg (function), 924
imf_descriptor (function), 1787
imf_energy_input_rate_noninstantaneous (function), 1787
imf_index_lookup (function), 1799
imf_maximum_mass (function), 1787
imf_metal_yield_rate_noninstantaneous (function), 1787
imf_minimum_mass (function), 1788
imf_name (function), 1789
imf_phi (function), 1789
imf_recycled_fraction_instantaneous (function), 1789
imf_recycling_rate_noninstantaneous (function), 1789
imf_select (function), 1789
imf_select_disk_spheroid (function), 1798
imf_select_disk_spheroid_initialize (subroutine), 1798
imf_select_fixed (function), 1798
imf_select_fixed_initialize (subroutine), 1798
imf_tabulate (subroutine), 1789
imf_yield_instantaneous (function), 1790
importerunitconvert (interface), 1254
importerunitconvert1d (function), 1254
importerunitconvert2d (function), 1255
importerunitconvertscalar (function), 1255
importerunits (type), 1255
importerunitsexponentiate (function), 1255
importerunitsmultiply (function), 1255
inc_gam.F90 (file), 1181
incomplete_gamma (module), 1181
index (interface), 1216
index_ch_vs (function), 1216
index_vs_ch (function), 1216
index_vs_vs (function), 1216
indicesbranchtip (function), 1450
indicesbranchtipattribute_match (function), 1450
indicesbranchtipisgettable (function), 1450
indicescountlinked (function), 1450
indicescreatebyinterrupt (subroutine), 1450
indicescreatelinked (subroutine), 1450
indicesdestroylinked (subroutine), 1451
indicesget (function), 1451
CODE INDEX

indicesnullbindinginteger0in (function), 1451
indicesnullbindinglonginteger0inout (function), 1451
indicesnullbindingrateinteger0in (subroutine), 1451
indicesnullbindingratelonginteger0inout (subroutine), 1451
indicesnullbindingsetinteger0in (subroutine), 1451
indicesnullbindingsetlonginteger0inout (subroutine), 1451
indicesstandardbranchtipget (function), 1451
indicesstandardbranchtipset (subroutine), 1452
infall_radii_cooling_freefall (module), 968
infall_radii_cooling_radius (module), 969
infall_radius (function), 968
infall_radius_cooling_freefall (function), 968
infall_radius_cooling_freefall_initialize (subroutine), 968
infall_radius_cooling_radius (function), 969
infall_radius_cooling_radius_initialize (subroutine), 969
infall_radius_growth_rate (function), 968
infall_radius_growth_rate_cooling_freefall (function), 968
infall_radius_growth_rate_cooling_radius (function), 969
infall_radius_initialize (subroutine), 968
infinitedefaultconstructor (function), 1760, 1771
infinitedestructor (subroutine), 1760, 1771
infinitedestructor (function), 1760, 1771
initialize_cosmological_mass_variance (subroutine), 1869
initialize_display (subroutine), 1092
initialize_integer_scalar (subroutine), 1937
inoue2014apply (subroutine), 1836, 1839
inoue2014defaultconstructor (function), 1836, 1839
input_parameter_is_present (function), 1951
input_parameters (module), 1939
input_parameters_file_close (subroutine), 1951
input_parameters_file_open (subroutine), 1951
inputparameterlist (type), 1952
inputparameterlistadd (subroutine), 1952
inputparameterlistdestructor (subroutine), 1952
inputparameterlistoutputtoxml (subroutine), 1952
insert (interface), 1216
insert_ch_ch (function), 1216
insert_ch_vs (function), 1216
insert_vs_ch (function), 1216
insert_vs_vs (function), 1216
instruments.filters.F90 (file), 1183
instruments_filters (module), 1183
integer_buffer_dump (subroutine), 1124
integer_buffer_extend (subroutine), 1124
integrand1 (function), 1902
integrand2 (function), 1902
integrand3 (function), 1902
integrand4 (function), 1902
<table>
<thead>
<tr>
<th>Description</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>integrate (function)</td>
<td>1308</td>
</tr>
<tr>
<td>integrate_done (subroutine)</td>
<td>1308</td>
</tr>
<tr>
<td>integration_gsl_error_handler (subroutine)</td>
<td>1309</td>
</tr>
<tr>
<td>intergalactic_medium.state.F90 (file)</td>
<td>1184</td>
</tr>
<tr>
<td>intergalactic_medium.state.file.F90 (file)</td>
<td>1191</td>
</tr>
<tr>
<td>intergalactic_medium.state.internal.evolver.F90 (file)</td>
<td>1195</td>
</tr>
<tr>
<td>intergalactic_medium.state.internal.F90 (file)</td>
<td>1193</td>
</tr>
<tr>
<td>intergalactic_medium.state.RecFast.F90 (file)</td>
<td>1191</td>
</tr>
<tr>
<td>intergalactic_medium.state.simple.F90 (file)</td>
<td>1196</td>
</tr>
<tr>
<td>intergalactic_medium.state (module)</td>
<td>1184</td>
</tr>
<tr>
<td>intergalactic_medium_state_internal_evolver (module)</td>
<td>1195</td>
</tr>
<tr>
<td>intergalactic_medium_state_internal_initialize (subroutine)</td>
<td>1195</td>
</tr>
<tr>
<td>intergalactic_medium_state_internal_odes (function)</td>
<td>1195</td>
</tr>
<tr>
<td>intergalactic_medium_state_internal_update (function)</td>
<td>1196</td>
</tr>
<tr>
<td>intergalacticmediumstate (interface)</td>
<td>1185</td>
</tr>
<tr>
<td>intergalacticmediumstateclass (type)</td>
<td>1185</td>
</tr>
<tr>
<td>intergalacticmediumstateconstructordefault (function)</td>
<td>1185</td>
</tr>
<tr>
<td>intergalacticmediumstateconstructornamed (function)</td>
<td>1186</td>
</tr>
<tr>
<td>intergalacticmediumstatedoublyionizedheliumfraction (function)</td>
<td>1186</td>
</tr>
<tr>
<td>intergalacticmediumstateelectronfractionnull (function)</td>
<td>1186</td>
</tr>
<tr>
<td>intergalacticmediumstateelectronscatteringintegrand (function)</td>
<td>1186</td>
</tr>
<tr>
<td>intergalacticmediumstateelectronscatteringopticaldepth (function)</td>
<td>1186</td>
</tr>
<tr>
<td>intergalacticmediumstateelectronscatteringtabulate (subroutine)</td>
<td>1186</td>
</tr>
<tr>
<td>intergalacticmediumstateelectronscatteringtime (function)</td>
<td>1186</td>
</tr>
<tr>
<td>intergalacticmediumstatefile (interface)</td>
<td>1186, 1193</td>
</tr>
<tr>
<td>intergalacticmediumstateinitialize (subroutine)</td>
<td>1187</td>
</tr>
<tr>
<td>intergalacticmediumstateinternal (interface)</td>
<td>1187, 1193</td>
</tr>
<tr>
<td>intergalacticmediumstateisfinalizable (function)</td>
<td>1187</td>
</tr>
<tr>
<td>intergalacticmediumstateneutralheliumfractionnull (function)</td>
<td>1187</td>
</tr>
<tr>
<td>intergalacticmediumstateneutralhydrogenfractionnull (function)</td>
<td>1187</td>
</tr>
<tr>
<td>intergalacticmediumstaterexcfast (interface)</td>
<td>1187, 1191</td>
</tr>
<tr>
<td>intergalacticmediumstatesimple (interface)</td>
<td>1187, 1197</td>
</tr>
<tr>
<td>intergalacticmediumstatesinglyionizedheliumfractionnull (function)</td>
<td>1187</td>
</tr>
<tr>
<td>intergalacticmediumstatesinglyionizedhydrogenfraction (function)</td>
<td>1188</td>
</tr>
<tr>
<td>intergalacticmediumstatetemperaturenull (function)</td>
<td>1188</td>
</tr>
<tr>
<td>internaldefaultconstructor (function)</td>
<td>1188, 1193</td>
</tr>
<tr>
<td>internaldensityhelium1set (subroutine)</td>
<td>1188, 1193</td>
</tr>
<tr>
<td>internaldensityhelium2set (subroutine)</td>
<td>1188, 1193</td>
</tr>
<tr>
<td>internaldensityhelium3set (subroutine)</td>
<td>1188, 1193</td>
</tr>
<tr>
<td>internaldensityhydrogen1set (subroutine)</td>
<td>1188, 1193</td>
</tr>
<tr>
<td>internaldensityhydrogen2set (subroutine)</td>
<td>1188, 1193</td>
</tr>
<tr>
<td>internalelectronfraction (function)</td>
<td>1189, 1194</td>
</tr>
<tr>
<td>internalfilteringmass (function)</td>
<td>1189, 1194</td>
</tr>
<tr>
<td>internalmassfilteringset (subroutine)</td>
<td>1189, 1194</td>
</tr>
<tr>
<td>internalneutralheliumfraction (function)</td>
<td>1189, 1194</td>
</tr>
<tr>
<td>internalneutralhydrogenfraction (function)</td>
<td>1189, 1194</td>
</tr>
<tr>
<td>internalsinglyionizedheliumfraction (function)</td>
<td>1189, 1194</td>
</tr>
<tr>
<td>internaltemperature (function)</td>
<td>1189, 1194</td>
</tr>
<tr>
<td>internaltemperaturereset (subroutine)</td>
<td>1190, 1195</td>
</tr>
</tbody>
</table>
CODE INDEX

internaltimeset (subroutine), 1190, 1195
interoutputcountlinked (function), 1452
interoutputcreatebyinterrupt (subroutine), 1452
interoutputcreateelinked (subroutine), 1452
interoutputdestroylinked (subroutine), 1452
interoutputdiskstarformationrate (function), 1452
interoutputdiskstarformationrateattributematch (function), 1452
interoutputdiskstarformationratesetisgettable (function), 1452
interoutputdiskstarformationraterate (subroutine), 1453
interoutputget (function), 1453
interoutputnullbindingdouble0inout (function), 1453
interoutputnullbindinginteger0in (function), 1453
interoutputnullbindingratedouble0inout (subroutine), 1453
interoutputnullbindingratedouble0inout (subroutine), 1453
interoutputnullbindingrateinteger0in (subroutine), 1453
interoutputnullbindingsetdouble0inout (subroutine), 1453
interoutputnullbindingsetinteger0in (subroutine), 1453
interoutputspheroidstarformationrate (function), 1453
interoutputspheroidstarformationrateattributematch (function), 1454
interoutputspheroidstarformationratesetisgettable (function), 1454
interoutputspheroidstarformationraterate (subroutine), 1454
interoutputstandarddiskstarformationratecount (function), 1454
interoutputstandarddiskstarformationrateget (function), 1454
interoutputstandarddiskstarformationraterate (subroutine), 1454
interoutputstandarddiskstarformationratescale (subroutine), 1454
interoutputstandarddiskstarformationrateset (subroutine), 1454
interoutputstandardspheroidstarformationratecount (function), 1455
interoutputstandardspheroidstarformationrateget (function), 1455
interoutputstandardspheroidstarformationraterate (subroutine), 1455
interoutputstandardspheroidstarformationratescale (subroutine), 1455
interoutputstandardspheroidstarformationrateset (subroutine), 1455
interp2dirregularobject (type), 1309
interpolate (function), 1312
interpolate_2d_irregular (interface), 1309
interpolate_2d_irregular_array (function), 1309
interpolate_2d_irregular_scalar (function), 1309
interpolate_derivative (function), 1312
interpolate_done (subroutine), 1312
interpolate_in_wavenumber (subroutine), 1866
interpolate_linear_do (function), 1312
interpolate_linear_generate_factors (function), 1312
interpolate_linear_generate_gradient_factors (function), 1312
interpolate_locate (function), 1313
inverse_gamma_function_incomplete (function), 1228
inversegamma_function_incomplete_complementary (function), 1228
inversecosineintegral (function), 1127
io_hdf5 (module), 1916
io_hdf5_assert_attribute_type (subroutine), 1917
io_hdf5_assert_dataset_type (subroutine), 1918
io_hdf5_character_types (function), 1918
io_hdf5_close (subroutine), 1918
io_hdf5_create_reference_scalar_to_1d (subroutine), 1918
io_hdf5_create_reference_scalar_to_2d (subroutine), 1918
io_hdf5_create_reference_scalar_to_3d (subroutine), 1918
io_hdf5_create_reference_scalar_to_4d (subroutine), 1918
io_hdf5_create_reference_scalar_to_5d (subroutine), 1918
io_hdf5_dataset_rank (function), 1919
io_hdf5_dataset_size (function), 1919
io_hdf5_destroy (subroutine), 1919
io_hdf5_has_attribute (function), 1919
io_hdf5_has_dataset (function), 1919
io_hdf5_has_group (function), 1919
io_hdf5_initialize (subroutine), 1919
io_hdf5_is_open (function), 1919
io_hdf5_is_reference (function), 1919
io_hdf5_open_attribute (function), 1920
io_hdf5_open_dataset (function), 1920
io_hdf5_open_file (subroutine), 1920
io_hdf5_open_group (function), 1920
io_hdf5_path_to (function), 1920
io_hdf5_read_attribute_character_1d_array_allocatable (subroutine), 1920
io_hdf5_read_attribute_character_1d_array_static (subroutine), 1920
io_hdf5_read_attribute_character_scalar (subroutine), 1921
io_hdf5_read_attribute_double_1d_array_allocatable (subroutine), 1921
io_hdf5_read_attribute_double_1d_array_static (subroutine), 1921
io_hdf5_read_attribute_double_scalar (subroutine), 1921
io_hdf5_read_attribute_integer8_1d_array_allocatable (subroutine), 1921
io_hdf5_read_attribute_integer8_1d_array_static (subroutine), 1921
io_hdf5_read_attribute_integer8_scalar (subroutine), 1921
io_hdf5_read_attribute_integer_1d_array_allocatable (subroutine), 1921
io_hdf5_read_attribute_integer_1d_array_static (subroutine), 1922
io_hdf5_read_attribute_integer_scalar (subroutine), 1922
io_hdf5_read_attribute_varstring_1d_array_allocatable (subroutine), 1922
io_hdf5_read_attribute_varstring_1d_array_static_do_read (subroutine), 1922
io_hdf5_read_attribute_varstring_1d_array_allocatable_do_read (subroutine), 1922
io_hdf5_read_attribute_varstring_1d_array_static (subroutine), 1922
io_hdf5_read_attribute_varstring_scalar (subroutine), 1922
io_hdf5_read_attribute_varstring_scalar_do_read (subroutine), 1922
io_hdf5_read_dataset_character_1d_array_allocatable (subroutine), 1923
io_hdf5_read_dataset_character_1d_array_static (subroutine), 1923
io_hdf5_read_dataset_double_1d_array_allocatable (subroutine), 1923
io_hdf5_read_dataset_double_1d_array_static (subroutine), 1923
io_hdf5_read_dataset_double_2d_array_allocatable (subroutine), 1923
io_hdf5_read_dataset_double_2d_array_static (subroutine), 1923
io_hdf5_read_dataset_double_3d_array_allocatable (subroutine), 1923
io_hdf5_read_dataset_double_3d_array_static (subroutine), 1923
io_hdf5_read_dataset_double_4d_array_allocatable (subroutine), 1924
io_hdf5_read_dataset_double_4d_array_static (subroutine), 1924
io_hdf5_read_dataset_double_5d_array_allocatable (subroutine), 1924
CODE INDEX

io_hdf5_read_dataset_double_5d_array_static (subroutine), 1924
io_hdf5_read_dataset_integer8_1d_array_allocatable (subroutine), 1924
io_hdf5_read_dataset_integer8_1d_array_static (subroutine), 1924
io_hdf5_read_dataset_integer_1d_array_allocatable (subroutine), 1924
io_hdf5_read_dataset_integer_1d_array_static (subroutine), 1924
io_hdf5_read_dataset_varstring_1d_array_allocatable (subroutine), 1925
io_hdf5_read_dataset_varstring_1d_array_allocatable_do_read (subroutine), 1925
io_hdf5_read_dataset_varstring_1d_array_static (subroutine), 1925
io_hdf5_read_dataset_varstring_1d_array_static_do_read (subroutine), 1925
io_hdf5_set_defaults (subroutine), 1925
io_hdf5_uninitialize (subroutine), 1925
io_hdf5_write_attribute_character_1d (subroutine), 1925
io_hdf5_write_attribute_character_scalar (subroutine), 1925
io_hdf5_write_attribute_double_1d (subroutine), 1926
io_hdf5_write_attribute_double_scalar (subroutine), 1926
io_hdf5_write_attribute_integer8_1d (subroutine), 1926
io_hdf5_write_attribute_integer8_scalar (subroutine), 1926
io_hdf5_write_attribute_integer_1d (subroutine), 1926
io_hdf5_write_attribute_integer_scalar (subroutine), 1926
io_hdf5_write_attribute_varstring_1d (subroutine), 1926
io_hdf5_write_attribute_varstring_scalar (subroutine), 1926
io_hdf5_write_dataset_character_1d (subroutine), 1927
io_hdf5_write_dataset_double_1d (subroutine), 1927
io_hdf5_write_dataset_double_2d (subroutine), 1927
io_hdf5_write_dataset_double_3d (subroutine), 1927
io_hdf5_write_dataset_double_4d (subroutine), 1927
io_hdf5_write_dataset_double_5d (subroutine), 1927
io_hdf5_write_dataset_integer8_1d (subroutine), 1927
io_hdf5_write_dataset_integer_1d (subroutine), 1927
io_hdf5_write_dataset_varstring_1d (subroutine), 1928
io_hdf_assert_is_initialized (subroutine), 1928
io_xml (module), 1928
ipmpar (function), 947
is_subhalo_subhalo_merger (function), 1248
iso_varying_string (module), 1198
iso_varying_string.F90 (file), 1197
isothermalcalculationreset (subroutine), 1034, 1046
isothermalcircularvelocity (function), 1034, 1046
isothermalcircularvelocitymaximum (function), 1034, 1046
isothermalconstructor (function), 1034, 1046
isothermaldefaultconstructor (function), 1034, 1046
isothermaldensity (function), 1034, 1046
isothermaldestructor (subroutine), 1034, 1046
isothermalenclosedmass (function), 1034, 1047
isothermalenergy (function), 1035, 1047
isothermalenergygrowthrate (function), 1035, 1047
isothermalfreefallradius (function), 1035, 1047
isothermalfreefallradiusincreaserate (function), 1035, 1047
isothermalkspase (function), 1035, 1047

2110
isothermalpotential (function), 1035, 1048
isothermalradiusfromspecificangularmomentum (function), 1036, 1048
isothermalrotationnormalization (function), 1036, 1048

jiang2008constructor (function), 1760, 1767
jiang2008defaultconstructor (function), 1760, 1767
jiang2008destructor (subroutine), 1761, 1767
jiang2008staterestore (subroutine), 1761, 1767
jiang2008statesnapshot (subroutine), 1761, 1767
jiang2008statestore (subroutine), 1761, 1767
jiang2008timeuntilmerging (function), 1761, 1767

kepler_orbits (module), 1344
kepler_orbits_angular_momentum (function), 1345
kepler_orbits_angular_momentum_set (subroutine), 1345
kepler_orbits_apocenter_radius (function), 1345
kepler_orbits_apocenter_radius_set (subroutine), 1346
kepler_orbits_assert_is_defined (subroutine), 1346
kepler_orbits_builder (subroutine), 1346
kepler_orbits_destroy (subroutine), 1346
kepler_orbits_dump (subroutine), 1346
kepler_orbits_dump_raw (subroutine), 1346
kepler_orbits_eccentricity (function), 1346
kepler_orbits_eccentricity_set (subroutine), 1346
kepler_orbits_energy (function), 1346
kepler_orbits_energy_set (subroutine), 1347
kepler_orbits_host_mass (function), 1347
kepler_orbits_is_bound (function), 1347
kepler_orbits_is_defined (function), 1347
kepler_orbits_masses_set (subroutine), 1347
kepler_orbits_output (subroutine), 1347
kepler_orbits_output_count (subroutine), 1347
kepler_orbits_output_names (subroutine), 1347
kepler_orbits_pericenter_radius (function), 1348
kepler_orbits_pericenter_radius_set (subroutine), 1348
kepler_orbits_propagate (subroutine), 1348
kepler_orbits_radius (function), 1348
kepler_orbits_radius_set (subroutine), 1348
kepler_orbits_read_raw (subroutine), 1348
kepler_orbits_reset (subroutine), 1348
kepler_orbits_semi_major_axis (function), 1348
kepler_orbits_semi_major_axis_set (subroutine), 1348
kepler_orbits_specific_reduced_mass (function), 1349
kepler_orbits_velocity_radial (function), 1349
kepler_orbits_velocity_radial_set (subroutine), 1349
kepler_orbits_velocity_scale (function), 1349
kepler_orbits_velocity_tangential (function), 1349
kepler_orbits_velocity_tangential_set (subroutine), 1349
keplerorbit (type), 1349
key_integer_scalar_i (function), 1937
CODE INDEX

keys_integer_scalar (subroutine), 1937
kind_numbers (module), 1953
kitayamasuto1996defaultconstructor (function), 1888, 1891
kitayamasuto1996densitycontrast (function), 1888, 1891
kitayamasuto1996densitycontrastrateofchange (function), 1888, 1891
kmt09_molecular_fraction_fast (function), 1811
kmt09_molecular_fraction_slow (function), 1811
laceycole1993defaultconstructor (function), 1761, 1768
laceycole1993destructor (subroutine), 1761, 1768
laceycole1993timeuntilmerging (function), 1761, 1768
laceycole1993timeuntilmergingmassdependence (function), 1762, 1768
laceycole1993tormendefaultconstructor (function), 1762, 1768
laceycole1993tormendestructor (subroutine), 1762, 1769
laceycole1993tomenstaterestore (subroutine), 1762, 1769
laceycole1993tomenstatesnapshot (subroutine), 1762, 1769
laceycole1993tomenstatesstore (subroutine), 1762, 1769
laceycole1993tormentimeuntilmerging (function), 1762, 1769
last_host_descendent (function), 1248
len (interface), 1217
len_ (function), 1217
len_trim (interface), 1217
len_trim_ (function), 1217
lge (interface), 1217
lge_ch_vs (function), 1217
lge_vs_ch (function), 1217
lge_vs_vs (function), 1217
lg (interface), 1217
lg_ch_vs (function), 1217
lg_vs_ch (function), 1217
lg_vs_vs (function), 1217
linear_growth (module), 1865
linear_growth_factor (function), 1866
linear_growth_factor_logarithmic_derivative (function), 1866
linear_growth_factor_simple_tabulate (subroutine), 1867
linear_growth_initialize (subroutine), 1866
linear_growth_simple (module), 1867
linear_growth_simple_state_retrieve (subroutine), 1867
linear_growth_simple_state_store (subroutine), 1867
linear_growth_state_retrieve (subroutine), 1867
lle (interface), 1218
lle_ch_vs (function), 1218
lle_vs_ch (function), 1218
lle_vs_vs (function), 1218
lt (interface), 1218
lt_ch_vs (function), 1218
lt_vs_ch (function), 1218
lt_vs_vs (function), 1218
load_from_file_vs (subroutine), 1218
load_primus_mass_function (subroutine), 1119

2112
logarithmic_double_factorial (function), 1227
longintegerhistory (type), 1344
lss_integrand (function), 1818
luminositytable (type), 1828
lycsuppressapply (subroutine), 1836, 1840
lycsuppressdefaultconstructor (function), 1836, 1840
madau1995apply (subroutine), 1836, 1840
madau1995defaultconstructor (function), 1836, 1840
make_output_group (subroutine), 1124
make_parameters_group (subroutine), 1952
make_range (function), 1317
make_table (subroutine), 1877, 1879
make_variance_range (function), 1855
map_mass_alfalfa_hi_mass_function_z0_00 (function), 1120
map_radius_sdss_size_function_z0_07 (function), 1118
mapcomponentsdouble0 (function), 1455
mapcomponentsvoid (subroutine), 1455
mass_distribution_beta_profile_density (function), 1350
mass_distribution_beta_profile_density_gradient_radial (function), 1350
mass_distribution_beta_profile_density_radial_moment (function), 1350
mass_distribution_beta_profile_initialize (subroutine), 1350
mass_distribution_beta_profile_mass_enc_by_sphere (function), 1350
mass_distribution_beta_profile_potential (function), 1351
mass_distribution_create (function), 1351
mass_distribution_density_gradient_radial_null (function), 1351
mass_distribution_density_null (function), 1351
mass_distribution_density_radial_moment_null (function), 1351
mass_distribution_half_mass_radius_spherical (function), 1351
mass_distribution_hernquist_density (function), 1351
mass_distribution_hernquist_density_radial_moment (function), 1351
mass_distribution_hernquist_half_mass_radius (function), 1352
mass_distribution_hernquist_initialize (subroutine), 1352
mass_distribution_hernquist_mass_enc_by_sphere (function), 1352
mass_distribution_hernquist_potential (function), 1352
mass_distribution_is_dimensionless (function), 1352
mass_distribution_mass_enc_by_sphere_null (function), 1352
mass_distribution_mass_enc_by_sphere_spherical (function), 1352
mass_distribution_mass_enc_by_sphere_spherical_integrand (function), 1353
mass_distribution_nfw_density (function), 1353
mass_distribution_nfw_initialize (subroutine), 1353
mass_distribution_potential_null (function), 1353
mass_distribution_sersic_density (function), 1353
mass_distribution_sersic_density_radial_moment (function), 1353
mass_distribution_sersic_half_mass_radius (function), 1353
mass_distribution_sersic_half_mass_radius_projected (function), 1353
mass_distribution_sersic_initialize (subroutine), 1354
mass_distribution_sersic_mass_enc_by_sphere (function), 1354
mass_distribution_sersic_potential (function), 1354
mass_distribution_symmetry_cylindrical (function), 1354
mass_distribution_symmetry_none (function), 1354
mass_distribution_symmetry_spherical (function), 1354
mass_distributions (module), 1350
mass_error_alfalfa_hi_mass_function_z0_00 (function), 1120
mass_from_cosmological_root_variance (function), 1869
mass_function_covariance (program), 883
Mass_Function_Covariance.F90 (file), 883
mass_function_covariance_matrix (subroutine), 1818
mass_function_halo_mass_integrand (function), 882
mass_function_integrand_i (function), 1819
mass_function_sampling_integrand (function), 1240
mass_function_time_integrand (function), 882
mass_function_time_integrand_i (function), 1819
mass_function_time_normalization_integrand (function), 882
massdistribution (type), 1354
massdistributionbetaprofile (type), 1354
massdistributioncylindrical (type), 1354
massdistributionhernquist (type), 1355
massdistributionnfw (type), 1355
massdistributionsersic (type), 1355
massdistributionspherical (type), 1355
massflowstatisticscooledmass (function), 1455
massflowstatisticscooledmassattributematch (function), 1455
massflowstatisticscooledmassisgettable (function), 1456
massflowstatisticscountlinked (function), 1456
massflowstatisticscreatebyinterrupt (subroutine), 1456
massflowstatisticscreatelinked (subroutine), 1456
massflowstatisticsdestroylinked (subroutine), 1456
massflowstatisticsget (function), 1456
massflowstatisticsnullbindingdouble0inout (function), 1456
massflowstatisticsnullbindinginteger0in (function), 1456
massflowstatisticsnullbindingrangedouble0inout (subroutine), 1457
massflowstatisticsnullbindingrangerangeinteger0inout (subroutine), 1457
massflowstatisticsnullbindingsetdouble0inout (subroutine), 1457
massflowstatisticsnullbindingsetinteger0in (subroutine), 1457
massflowstatisticsstandardcooledmasscount (function), 1457
massflowstatisticsstandardcooledmassget (function), 1457
massflowstatisticsstandardcooledmassrate (subroutine), 1457
massflowstatisticsstandardcooledmassscale (subroutine), 1457
massflowstatisticsstandardcooledmassset (subroutine), 1457
massfunction (type), 1120
massfunctiondescriptor (type), 1120
massfunctionwork (type), 1120
massrootfunction (function), 1063
math.Bessel_functions.F90 (file), 1225
math.distributions.Gaussian.F90 (file), 1225
math.error_function.F90 (file), 1226
math.exponential_integrals.F90 (file), 1226
math.factorial.F90 (file), 1227

2114
math.gamma_function.F90 (file), 1227
math.hypergeometric_functions.F90 (file), 1228
math.trigonometric_functions.F90 (file), 1229
math.vector.F90 (file), 1230
math_distributions_gaussian (module), 1225
matterdarkenergyagetableodes (function), 982, 989
matterdarkenergyconstructor (function), 982, 989
matterdarkenergycosmictime (function), 982, 989
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergyqualityepochmatterdarkenergy (function), 983, 990
matterdarkenergyqualityepochmatterdarkenergy (function), 983, 990
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
matterdarkenergymakeexpansionfactortable (module), 1225
matterdarkenergymakeexpansionfactortable (function), 982, 989
matterdarkenergymakeexpansionfactortable (subroutine), 982, 989
CODE INDEX

matterlambdaomegadarkenergyepochal (function), 988, 995
matterlambdaomegammatterepochal (function), 988, 995
matterlambdaomegammatterrateofchange (function), 988, 995
matterlambdastatterestore (subroutine), 988, 995
matterlambdastatestore (subroutine), 988, 995
matterlambdatemperaturecmbepochal (function), 988, 995
matterlambdatimeatdistancecomoving (function), 988, 995
max (interface), 1329, 1725, 1736
mean_density_contrast_root (function), 1125
meiksin2006apply (subroutine), 1836, 1840
meiksin2006defaultconstructor (function), 1837, 1841
memory_management (module), 1955
memory_usage_get (function), 1974
memory_usage_record (subroutine), 1974
memory_usage_report (subroutine), 1974
memoryusage (type), 1974
memoryusagelist (type), 1974
merger_tree_branching (module), 1231
merger_tree_branching_modifier (function), 1236
merger_tree_branching_modifier_null (function), 1237
merger_tree_branching_modifier_parkinson (function), 1237
merger_tree_branching_modifiers (module), 1236
merger_tree_branching_modifiers_null (module), 1237
merger_tree_branching_modifiers_null_initialize (subroutine), 1237
merger_tree_branching_modifiers_parkinson (module), 1237
merger_tree_branching_modifiers_parkinson_initialize (subroutine), 1237
merger_tree_build (module), 1239
merger_tree_build_cole2000 (module), 1239
merger_tree_build_cole2000_initialize (subroutine), 1239
merger_tree_build_cole2000_snapshot (subroutine), 1239
merger_tree_build_cole2000_state_retrieve (subroutine), 1239
merger_tree_build_cole2000_state_store (subroutine), 1239
merger_tree_build_do (subroutine), 1240
merger_tree_build_do_cole2000 (subroutine), 1239
merger_tree_build_initialize (subroutine), 1240
merger_tree_build_mass_resolution (function), 1240
merger_tree_build_mass_resolution_fixed (function), 1241
merger_tree_build_mass_resolution_initialize (subroutine), 1241
merger_tree_build_mass_resolution_scaled (function), 1241
merger_tree_cladistic_information_content (function), 1362
merger_tree_conditional_mass_function (subroutine), 1238
merger_tree_conditional_mass_function_output (subroutine), 1238
merger_tree_construct_fully_specified (subroutine), 1242
merger_tree_construct_fully_specified_initialize (subroutine), 1242
merger_tree_construct_mass_function_sampling (function), 1243
merger_tree_construct_mass_function_sampling_gaussian (function), 1243
merger_tree_construct_mass_function_sampling_halo_mf (function), 1244
merger_tree_construct_mass_function_sampling_power_law (function), 1244
merger_tree_construct_mass_function_sampling_stellar_mf (function), 1245
<table>
<thead>
<tr>
<th>Code Index</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>merger_tree_construction (module), 1238</td>
<td></td>
</tr>
<tr>
<td>merger_tree_create (function), 1238</td>
<td></td>
</tr>
<tr>
<td>merger_tree_create_event (function), 1458</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_construct_particle_indices (subroutine), 1356</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_set_subhalo_masses (subroutine), 1356</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure (module), 1356</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_add_metadata (subroutine), 1356</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_add_metadata_double (subroutine), 1356</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_add_metadata_integer (subroutine), 1357</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_add_metadata_text (subroutine), 1357</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_export (subroutine), 1357</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_export_galacticus (subroutine), 1357</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_export_irate (subroutine), 1357</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_make_references (subroutine), 1357</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_read_ascii (subroutine), 1357</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_read_particles_ascii (subroutine), 1357</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_reset (subroutine), 1358</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_includes_hubble_flow (subroutine), 1358</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_includes_subhalo_masses (subroutine), 1358</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_is_periodic (subroutine), 1358</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_node_count (subroutine), 1358</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_particle_count (subroutine), 1358</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_particle_mass (subroutine), 1358</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_particle_property_column (subroutine), 1358</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_property_column (subroutine), 1359</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_property_double (subroutine), 1359</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_property_integer8 (subroutine), 1359</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_self_contained (subroutine), 1359</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_self_hosting_halo_id (subroutine), 1359</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_tree_count (subroutine), 1359</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_tree_indices (subroutine), 1359</td>
<td></td>
</tr>
<tr>
<td>merger_tree_data_structure_set_units (subroutine), 1359</td>
<td></td>
</tr>
<tr>
<td>merger_tree_validate_trees (subroutine), 1359</td>
<td></td>
</tr>
<tr>
<td>merger_tree_destroy (subroutine), 1458</td>
<td></td>
</tr>
<tr>
<td>merger_tree_destroy_branch (subroutine), 1458</td>
<td></td>
</tr>
<tr>
<td>merger_tree_dump (subroutine), 1360</td>
<td></td>
</tr>
<tr>
<td>merger_tree_dump_evolution (subroutine), 1274</td>
<td></td>
</tr>
<tr>
<td>merger_tree_dump_evolution_close (subroutine), 1274</td>
<td></td>
</tr>
<tr>
<td>merger_tree_earliest_time (function), 1458</td>
<td></td>
</tr>
<tr>
<td>merger_tree_earliest_time_evolving (function), 1458</td>
<td></td>
</tr>
<tr>
<td>merger_tree_evolve_to (subroutine), 1275</td>
<td></td>
</tr>
<tr>
<td>merger_tree_history_store (subroutine), 1279</td>
<td></td>
</tr>
<tr>
<td>merger_tree_history_write (subroutine), 1279</td>
<td></td>
</tr>
<tr>
<td>merger_tree_initialize (subroutine), 1284</td>
<td></td>
</tr>
<tr>
<td>merger_tree_latest_time (function), 1458</td>
<td></td>
</tr>
<tr>
<td>merger_tree_mass_accretion_history (module), 1284</td>
<td></td>
</tr>
<tr>
<td>merger_tree_mass_accretion_history_close (subroutine), 1285</td>
<td></td>
</tr>
<tr>
<td>merger_tree_mass_accretion_history_output (subroutine), 1285</td>
<td></td>
</tr>
</tbody>
</table>
merger_tree_monotonic_mass_growth (subroutine), 1285
merger_tree_node_get (function), 1458
merger_tree_non_essential_branches (subroutine), 1287
merger_tree_output_structure (module), 1285
merger_tree_prune_branches (subroutine), 1286
merger_tree_prune_hierarchy (subroutine), 1286
merger_tree_prune_lightcone (subroutine), 1128
merger_tree_read (module), 1245
merger_tree_read_do (subroutine), 1248
merger_tree_read_importers (module), 1251
merger_tree_read_initialize (subroutine), 1249
merger_tree_read_state (module), 1272
merger_tree_read_state_retrieve (subroutine), 1272
merger_tree_read_state_store (subroutine), 1272
merger_tree_record_evolution_output (subroutine), 1280
merger_tree_record_evolution_store (subroutine), 1280
merger_tree_regrid_time (subroutine), 1287
merger_tree_remove_event (subroutine), 1458
merger_tree_smooth_accretion (module), 1272
merger_tree_smooth_accretion_do (subroutine), 1272
merger_tree_smooth_accretion_initialize (subroutine), 1272
merger_tree_state_restore (subroutine), 1273
merger_tree_state_store (subroutine), 1273
merger_tree_state_store_initialize (subroutine), 1273
merger_tree_state_walk_tree (subroutine), 1273
merger_tree_structure_dump (subroutine), 1274
merger_tree_structure_output (subroutine), 1285
merger_tree_structure_preclose (subroutine), 1286
merger_tree_timestep_history (subroutine), 1279
merger_tree_timestep_record_evolution (subroutine), 1280
merger_tree_timestep_satellite (subroutine), 1281
merger_tree_timestep_simple (subroutine), 1282
merger_tree_timesteps (module), 1279
merger_tree_timesteps_history (module), 1279
merger_tree_timesteps_record_evolution (module), 1280
merger_tree_timesteps_satellite (module), 1281
merger_tree_timesteps_simple (module), 1282
merger_tree_walk_descend_to_progenitors (function), 1458
merger_tree_write (subroutine), 1288
merger_trees.branching_probability.F90 (file), 1231
merger_trees.branching_probability.generalized_Press_Schechter.F90 (file), 1232
merger_trees.branching_probability.modified_Press_Schechter.F90 (file), 1234
merger_trees.branching_probability.modifier.F90 (file), 1236
merger_trees.branching_probability.modifier.null.F90 (file), 1237
merger_trees.branching_probability.modifier.Parkinson.F90 (file), 1237
merger_trees.conditional_mass_functions.F90 (file), 1237
merger_trees.construct.build.F90 (file), 1239
merger_trees.construct.build.mass_resolution.F90 (file), 1240
merger_trees.construct.build.mass_resolution.fixed.F90 (file), 1241
merger_trees.construct.build.mass_resolution.scaled.F90 (file), 1241
merger_trees.construct.F90 (file), 1238
merger_trees.construct.fully_specified.F90 (file), 1242
merger_trees.construct.mass_function_sampling.F90 (file), 1243
merger_trees.construct.mass_function_sampling.gaussian.F90 (file), 1243
merger_trees.construct.mass_function_sampling.halo_mass_function.F90 (file), 1244
merger_trees.construct.mass_function_sampling.power_law.F90 (file), 1244
merger_trees.construct.mass_function_sampling.stellar_mass_function.F90 (file), 1245
merger_trees.construct.read.F90 (file), 1245
merger_trees.construct.read.importer.F90 (file), 1251
merger_trees.construct.read.importer.galacticus.F90 (file), 1268
merger_trees.construct.read.importer.SussingMergerTrees.F90 (file), 1263
merger_trees.construct.read.state.F90 (file), 1272
merger_trees.construct.smooth_accretion.F90 (file), 1272
merger_trees.construct.state_restore.F90 (file), 1273
merger_trees.dump_evolution.F90 (file), 1274
merger_trees.dump_structure.F90 (file), 1274
merger_trees.evolve.deadlock_options.F90 (file), 1276
merger_trees.evolve.F90 (file), 1274
merger_trees.evolve.node.F90 (file), 1276
merger_trees.evolve.timesteps.F90 (file), 1278
merger_trees.evolve.timesteps.history.F90 (file), 1279
merger_trees.evolve.timesteps.record_evolution.F90 (file), 1280
merger_trees.evolve.timesteps.report.F90 (file), 1281
merger_trees.evolve.timesteps.satellite.F90 (file), 1281
merger_trees.evolve.timesteps.simple.F90 (file), 1282
merger_trees.evolve.timesteps.template.F90 (file), 1282
merger_trees.file_maker.Bolshoi.F90 (file), 1282
merger_trees.file_maker.Millennium.F90 (file), 1283
merger_trees.file_maker.simple.F90 (file), 1283
merger_trees.initialize.F90 (file), 1284
merger_trees.mass_accretion_history.F90 (file), 1284
merger_trees.monotonic_mass_growth.F90 (file), 1285
merger_trees.output_structure.F90 (file), 1285
merger_trees.prune_branches.F90 (file), 1286
merger_trees.prune_hierarchy.F90 (file), 1286
merger_trees.prune_non_essential.F90 (file), 1286
merger_trees.regrid_times.F90 (file), 1287
merger_trees.render.F90 (file), 1287
merger_trees.write.F90 (file), 1288
merger_trees_bolshoi (module), 1283
merger_trees_bolshoi_process (subroutine), 1283
merger_trees_build_mass_resolution (module), 1240
merger_trees_build_mass_resolution_fixed (module), 1241
merger_trees_build_mass_resolution_fixed_initialize (subroutine), 1241
merger_trees_build_mass_resolution_scaled (module), 1241
merger_trees_build_mass_resolution_scaled_initialize (subroutine), 1242
merger_trees_conditional_mass_function (module), 1238
merger_trees_construct_fully_specified (module), 1242
merger_trees_dump (module), 1360
merger_trees_dump_evolution (module), 1274
merger_trees_evolve (module), 1274
merger_trees_evolve_deadlock_status (module), 1276
merger_trees_evolve_node (module), 1276
merger_trees_evolve_timesteps_template (module), 1282
merger_trees_information_content (module), 1362
merger_trees_initialize (module), 1284
merger_trees_mass_function_sampling (module), 1243
merger_trees_mass_function_sampling_gaussian (module), 1243
merger_trees_mass_function_sampling_gaussian_initialize (subroutine), 1243
merger_trees_mass_function_sampling_halo_mf (module), 1244
merger_trees_mass_function_sampling_halo_mf_initialize (subroutine), 1244
merger_trees_mass_function_sampling_initialization (subroutine), 1243
merger_trees_mass_function_sampling_power_law (module), 1244
merger_trees_mass_function_sampling_power_law_initialize (subroutine), 1244
merger_trees_mass_function_sampling_stellar_mf (module), 1245
merger_trees_mass_function_sampling_stellar_mf_initialize (subroutine), 1245
merger_trees_millennium (module), 1283
merger_trees_millennium_process (subroutine), 1283
merger_trees_monotonic_mass_growth (module), 1285
merger_trees_prune_branches (module), 1286
merger_trees_prune_hierarchy (module), 1286
merger_trees_prune_non_essential (module), 1286
merger_trees_regrid_times (module), 1287
merger_trees_render (module), 1287
merger_trees_render_dump (subroutine), 1287
merger_trees_simple (module), 1283
merger_trees_simple_process (subroutine), 1283
merger_trees_state_store (module), 1273
merger_trees_write (module), 1288
mergertree (type), 1458
mergertreedata (type), 1360
mergertreeimporter (interface), 1255
mergertreeimporterangularmomenta3davailablenull (function), 1255
mergertreeimporterangularmomentaavailablenull (function), 1255
mergertreeimporterangularmomentaincludesubhalosnull (function), 1255
mergertreeimporterclass (type), 1256
mergertreeimporterclosenull (subroutine), 1256
mergertreeimporterconstructor defaulted (function), 1256
mergertreeimporterconstructornamed (function), 1256
mergertreeimportercubelengthnull (function), 1256
mergertreeimportergalacticus (interface), 1256, 1271
mergertreeimporterimportnull (subroutine), 1256
mergertreeimporterinitialize (subroutine), 1256
mergertreeimporterisfinalizable (function), 1256
mergertreeimportermassesincludesubhalosnull (function), 1257
mergertreeimporternodecountnull (function), 1257
mergetreeimporteropennonnull (subroutine), 1257
mergetreeimporterparticlecountavailablenonnull (function), 1257
mergetreeimporterpositionsareperiodicnonnull (function), 1257
mergetreeimporterpositionsavailablenonnull (function), 1257
mergetreeimporterscaleradiiavailablenonnull (function), 1257
mergetreeimporterspin3davailablenonnull (function), 1257
mergetreeimporterspinavailablenonnull (function), 1258
mergetreeimportersubhalotracecountnonnull (function), 1258
mergetreeimportersubhalotracenonnull (subroutine), 1258
mergetreeimportersussing (interface), 1258, 1264
mergetreeimportertreecountnonnull (function), 1258
mergetreeimportertreeindexnonnull (function), 1258
mergetreeimportertreesareselfcontainednonnull (function), 1258
mergetreeimportertreeseshavesubhalosnonnull (function), 1258
mergetreeimportertreeweightnonnull (function), 1259
mergetreeimportervelocitiesincludehubbleflownonnull (function), 1259
mergetreeimportervelocitydispersionavailablenonnull (function), 1259
mergetreeimportervelocitymaximumavailablenonnull (function), 1259
mergertreelast (type), 1459
merging_rate (function), 1234, 1235
mergingstatisticscountlinked (function), 1459
mergingstatisticsscreatebyinterrupt (subroutine), 1459
mergingstatisticsscreateelinked (subroutine), 1459
mergingstatisticsdestroylinked (subroutine), 1459
mergingstatisticsgalaxymajormergertime (function), 1459
mergingstatisticsgalaxymajormergertimeattributematch (function), 1459
mergingstatisticsgalaxymajormergertimeisgettable (function), 1459
mergingstatisticsget (function), 1459
mergingstatisticsnodeformationtime (function), 1460
mergingstatisticsnodeformationtimeattributematch (function), 1460
mergingstatisticsnodeformationtimeisgettable (function), 1460
mergingstatisticsnodehierarchylevel (function), 1460
mergingstatisticsnodehierarchylevelattributematch (function), 1460
mergingstatisticsnodehierarchylevelisgettable (function), 1460
mergingstatisticsnodemajormergertime (function), 1460
mergingstatisticsnodemajormergertimeattributematch (function), 1460
mergingstatisticsnodemajormergertimeisgettable (function), 1461
mergingstatisticsnullbindingdouble0inout (function), 1461
mergingstatisticsnullbindinginteger0in (function), 1461
mergingstatisticsnullbindinginteger0inout (function), 1461
mergingstatisticsnullbindinginteger1inout (function), 1461
mergingstatisticsnullbindingratedouble0inout (subroutine), 1461
mergingstatisticsnullbindingrateinteger0in (subroutine), 1461
mergingstatisticsnullbindingrateinteger0inout (subroutine), 1461
mergingstatisticsnullbindingrateinteger1inout (subroutine), 1461
mergingstatisticsnullbindingsetdouble0inout (subroutine), 1462
mergingstatisticsnullbindingsetinteger0in (subroutine), 1462
mergingstatisticsnullbindingsetinteger0inout (subroutine), 1462
mergingstatisticsnullbindingsetinteger1inout (subroutine), 1462
mergingstatisticsrecentmajormergercount (function), 1462
mergingstatisticsrecentmajormergercountattributematch (function), 1462
mergingstatisticsrecentmajormergercountisgettable (function), 1462
mergingstatisticsrecentmajormergercountget (function), 1462
mergingstatisticsrecentmajormergercountset (subroutine), 1462
mergingstatisticsstandardgalaxymajormergergetimeget (function), 1463
mergingstatisticsstandardgalaxymajormergergetimeset (subroutine), 1463
mergingstatisticsstandardnodeformationtimeget (function), 1463
mergingstatisticsstandardnodeformationtimeset (subroutine), 1463
mergingstatisticsstandardnodehierarchylevelget (function), 1463
mergingstatisticsstandardnodehierarchylevelset (subroutine), 1463
mergingstatisticsstandardnodemajormergergetimeget (function), 1463
mergingstatisticsstandardnodemajormergergetimeset (subroutine), 1464
meshes (module), 1313
meshes_apply_point (subroutine), 1313
meta_tree_timing_initialize (subroutine), 1114
meta_tree_timing_output (subroutine), 1114
meta_tree_timing_post_evolve (subroutine), 1114
meta_tree_timing_pre_construction (subroutine), 1114
meta_tree_timing_pre_evolve (subroutine), 1115
metal_yield_integrand (function), 1790
millennium_merger_tree_file_maker (program), 883
Millennium_Merger_Tree_File_Maker.F90 (file), 883
modification_function (function), 1235
modified_press_schechter_branch_mass (function), 1235
modified_press_schechter_branch_mass_root (function), 1235
modified_press_schechter_branching (module), 1234
modified_press_schechter_branching_initialize (subroutine), 1235
modified_press_schechter_branching_maximum_step (function), 1235
modified_press_schechter_branching_probability (function), 1236
modified_press_schechter_subresolution_fraction (function), 1236
munozcuartas2011concentration (function), 1054, 1058
munozcuartas2011darkmatterprofiledefinition (function), 1054, 1058
munozcuartas2011defaultconstructor (function), 1054, 1059
munozcuartas2011densitycontrastdefinition (function), 1054, 1059
naozbarkana2007constructor (function), 892, 896
naozbarkana2007defaultconstructor (function), 892, 896
naozbarkana2007failedfraction (function), 892, 896
naozbarkana2007fromsimple (subroutine), 892, 896
nbodycountlinked (function), 1464
nbodycreatebyinterrupt (subroutine), 1464
nbodycreatelinked (subroutine), 1464
nbodydestroylinked (subroutine), 1464
nbodyget (function), 1464
nbodynullbindingdouble0inout (function), 1464
nbodynullbindinginteger0in (function), 1464
nbodynullbindinglonginteger0inout (function), 1464
nbodynullbindingratedouble0inout (subroutine), 1465
nbodynullbindingrateinteger0in (subroutine), 1465

2122
nbodynullbindingratelonginteger0inout (subroutine), 1465
nbodynullbindingsetdouble0inout (subroutine), 1465
nbodynullbindingsetinteger0in (subroutine), 1465
nbodynullbindingsetlonginteger0inout (subroutine), 1465
nbodyparticlecount (function), 1465
nbodyparticlecountattributematch (function), 1465
nbodyparticlecountisgettable (function), 1465
nbodystandardparticlecountget (function), 1466
nbodystandardparticlecountset (subroutine), 1466
nbodystandardvelocitydispersionget (function), 1466
nbodystandardvelocitydispersionset (subroutine), 1466
nbodystandardvelocitymaximumget (function), 1466
nbodystandardvelocitymaximumset (subroutine), 1466
nbodyvelocitydispersion (function), 1466
nbodyvelocitydispersionattributematch (function), 1466
nbodyvelocitydispersionisgettable (function), 1466
nbodyvelocitymaximum (function), 1467
nbodyvelocitymaximumattributematch (function), 1467
nbodyvelocitymaximumisgettable (function), 1467
nfw1996concentration (function), 1055, 1059
nfw1996constructor (function), 1055, 1059
nfw1996darkmatterprofiledefinition (function), 1055, 1059
nfw1996defaultconstructor (function), 1055, 1059
nfw1996densitycontrastdefinition (function), 1055, 1060
nfw1996rootfunction (function), 1055, 1060
nfwangularmomentumscalefree (function), 1036, 1040
nfwcalculationreset (subroutine), 1036, 1040
nfwcircularenergy (function), 1036, 1041
nfwcircularenergyisgettable (function), 1036, 1041
nfwconstructor (function), 1037, 1041
nfwdefaultconstructor (function), 1037, 1041
nfwdensity (function), 1037, 1041
nfwdensityscalefree (function), 1037, 1041
nfwdestroyer (subroutine), 1037, 1041
nfwenclosedmass (function), 1037, 1041
nfwenclosedmassscalefree (function), 1037, 1042
nfwenergy (function), 1037, 1042
nfwenergyisgettable (function), 1038, 1042
nfwfreefallradius (function), 1038, 1042
nfwfreefallradiusincreaserate (function), 1038, 1042
nfwfreefalltabulate (subroutine), 1038, 1042
nfwfreefalltimescalefree (function), 1038, 1042
nfwfreefalltimescalefreescalefreeintegrand (function), 1038, 1042
nfwinverseangularmomentum (subroutine), 1038, 1043
nfwjeansequationintegrand (function), 1039, 1043
nfwkineticenergyintegrand (function), 1039, 1043
nfwkspace (function), 1038, 1043
nfwpotential (function), 1039, 1043
nfwpotentialenergyintegrand (function), 1039, 1043
CODE INDEX

nfwprofileenergy (function), 1039, 1043
nfwradiusfromspecificangularmomentum (function), 1039, 1043
nfwrotationnormalization (function), 1039, 1045
nfwspecificangularmomentumscalefree (function), 1040, 1045
nfwstaterestore (subroutine), 1040, 1045
nfwstatestore (subroutine), 1040, 1045
nfwtabulate (subroutine), 1040, 1045
node_array_position (function), 1273
node_branch_jump (function), 1066
node_branch_jumps (module), 1065
node_component_age_statistics_standard (module), 1678
node_component_age_statistics_standard_rate_compute (subroutine), 1678
node_component_age_statistics_standard_satellite_merger (subroutine), 1678
node_component_age_statistics_standard_scale_set (subroutine), 1678
node_component_age_statistics_builder (subroutine), 1467
node_component_age_statistics_destroy (subroutine), 1467
node_component_age_statistics_dump (subroutine), 1467
node_component_age_statistics_initializer (subroutine), 1467
node_component_age_statistics_move (subroutine), 1467
node_component_age_statistics_output (subroutine), 1468
node_component_age_statistics_output_count (subroutine), 1468
node_component_age_statistics_output_names (subroutine), 1468
node_component_age_statistics_remove (subroutine), 1468
node_component_age_statistics_type (function), 1468
node_component_age_statisticsnull_builder (subroutine), 1468
node_component_age_statisticsnull_count (function), 1468
node_component_age_statisticsnull_deserialize_rates (subroutine), 1468
node_component_age_statisticsnull_deserialize_scales (subroutine), 1468
node_component_age_statisticsnull_deserialize_values (subroutine), 1469
node_component_age_statisticsnull_destroy (subroutine), 1469
node_component_age_statisticsnull_dump (subroutine), 1469
node_component_age_statisticsnull_dump_raw (subroutine), 1469
node_component_age_statisticsnull_dump_xml (subroutine), 1469
node_component_age_statisticsnull_initializer (subroutine), 1469
node_component_age_statisticsnull_is_active (function), 1469
node_component_age_statisticsnull_name_from_index (subroutine), 1469
node_component_age_statisticsnull_ode_step_rates_init (subroutine), 1469
node_component_age_statisticsnull_odestepscalesinit (subroutine), 1470
node_component_age_statisticsnull_output_count (subroutine), 1470
node_component_age_statisticsnull_output_names (subroutine), 1470
node_component_age_statisticsnull_read_raw (subroutine), 1470
node_component_age_statisticsnull_serialize_rates (subroutine), 1470
node_component_age_statisticsnull_serialize_scales (subroutine), 1470
node_component_age_statisticsnull_serialize_values (subroutine), 1470
node_component_age_statisticsnull_type (function), 1470
node_component_age_statisticsstandard_builder (subroutine), 1470
node_component_age_statisticsstandard_count (function), 1471
node_component_age_statisticsstandard_deserialize_rates (subroutine), 1471
node_component_age_statisticsstandard_deserialize_scales (subroutine), 1471
node_component_agestatisticsstandard_deserialize_values (subroutine), 1471
node_component_agestatisticsstandard_destroy (subroutine), 1471
node_component_agestatisticsstandard_dump (subroutine), 1471
node_component_agestatisticsstandard_dump_raw (subroutine), 1471
node_component_agestatisticsstandard_dump_xml (subroutine), 1471
node_component_agestatisticsstandard_initialize (subroutine), 1471
node_component_agestatisticsstandard_is_active (function), 1472
node_component_agestatisticsstandard_name_from_index (subroutine), 1472
node_component_agestatisticsstandard_ode_step_rates_init (subroutine), 1472
node_component_agestatisticsstandard_odestepscalesinit (subroutine), 1472
node_component_agestatisticsstandard_output_count (subroutine), 1472
node_component_agestatisticsstandard_output_names (subroutine), 1472
node_component_agestatisticsstandard_read_raw (subroutine), 1472
node_component_agestatisticsstandard_serialize_rates (subroutine), 1472
node_component_agestatisticsstandard_serialize_scales (subroutine), 1473
node_component_agestatisticsstandard_serialize_values (subroutine), 1473
node_component_agestatisticsstandard_type (function), 1473
node_component_assign (subroutine), 1473
node_component_basic_builder (subroutine), 1473
node_component_basic_destroy (subroutine), 1473
node_component_basic_dump (subroutine), 1473
node_component_basic_initializer (subroutine), 1473
node_component_basic_move (subroutine), 1473
node_component_basic_non_evolving (module), 1679
node_component_basic_non_evolving_promote (subroutine), 1679
node_component_basic_non_evolving_rate_compute (subroutine), 1679
node_component_basic_non_evolving_scale_set (subroutine), 1679
node_component_basic_output (subroutine), 1474
node_component_basic_output_count (subroutine), 1474
node_component_basic_output_names (subroutine), 1474
node_component_basic_remove (subroutine), 1474
node_component_basic_standard (module), 1679
node_component_basic_standard_promote (subroutine), 1679
node_component_basic_standard_rate_compute (subroutine), 1680
node_component_basic_standard_scale_set (subroutine), 1680
node_component_basic_standard_stop_accretion (subroutine), 1680
node_component_basic_standard_tracking (module), 1680
node_component_basic_standard_tracking_promote (subroutine), 1680
node_component_basic_standard_tree_initialize (subroutine), 1680
node_component_basic_standard_tree_tracking_initialize (subroutine), 1680
node_component_basic_standard_unresolved_mass (function), 1680
node_component_basic_type (function), 1474
node_component_basicnonevolving_builder (subroutine), 1474
node_component_basicnon evolving_count (function), 1474
node_component_basicnon evolving_deserialize_rates (subroutine), 1474
node_component_basicnon evolving_deserialize_scales (subroutine), 1474
node_component_basicnon evolving_deserialize_values (subroutine), 1475
node_component_basicnon evolving_destroy (subroutine), 1475
node_component_basicnon evolving_dump (subroutine), 1475
node_component_basicstandard_output_count (subroutine), 1480
node_component_basicstandard_output_names (subroutine), 1481
node_component_basicstandard_read_raw (subroutine), 1481
node_component_basicstandard_serialize_rates (subroutine), 1481
node_component_basicstandard_serialize_scales (subroutine), 1481
node_component_basicstandard_serialize_values (subroutine), 1481
node_component_basicstandard_type (function), 1481
node_component_basicstandardtracking_builder (subroutine), 1481
node_component_basicstandardtracking_count (function), 1481
node_component_basicstandardtracking_deserialize_rates (subroutine), 1481
node_component_basicstandardtracking_deserialize_scales (subroutine), 1482
node_component_basicstandardtracking_deserialize_values (subroutine), 1482
node_component_basicstandardtracking_destroy (subroutine), 1482
node_component_basicstandardtracking_dump (subroutine), 1482
node_component_basicstandardtracking_dump_raw (subroutine), 1482
node_component_basicstandardtracking_dump_xml (subroutine), 1482
node_component_basicstandardtracking_initializor (subroutine), 1482
node_component_basicstandardtracking_is_active (function), 1482
node_component_basicstandardtracking_name_from_index (subroutine), 1482
node_component_basicstandardtracking_ode_step_rates_init (subroutine), 1483
node_component_basicstandardtracking_odestepscalesinit (subroutine), 1483
node_component_basicstandardtracking_output_count (subroutine), 1483
node_component_basicstandardtracking_output_names (subroutine), 1483
node_component_black_hole_simple (module), 1681
node_component_black_hole_simple_create (subroutine), 1681
node_component_black_hole_simple_enclosed_mass (function), 1484
node_component_black_hole_simple_initialize (subroutine), 1681
node_component_black_hole_simple_matches (function), 1681
node_component_black_hole_simple_output (subroutine), 1681
node_component_black_hole_simple_output_count (subroutine), 1681
node_component_black_hole_simple_output_names (subroutine), 1681
node_component_black_hole_simple_potential (function), 1682
node_component_black_hole_simple_rate_compute (subroutine), 1682
node_component_black_hole_simple_rotation_curve (function), 1682
node_component_black_hole_simple_rotation_curve_gradient (function), 1683
node_component_black_hole_simple_scale_set (subroutine), 1682
node_component_black_hole_simple_seed_mass (function), 1484
node_component_black_hole_simple_structure (module), 1682
node_component_black_hole_standard (module), 1683
node_component_black_hole_standard_accretion_rate (function), 1683
node_component_black_hole_standard_create (subroutine), 1683
node_component_black_hole_standard_enclosed_mass (function), 1484
node_component_black_hole_standard_initialize (subroutine), 1683
node_component_black_hole_standard_mass_accretion_rate (subroutine), 1684
node_component_black_hole_standard_matches (function), 1684
node_component_black_hole_standard_merge_black_holes (subroutine), 1684
node_component_black_hole_standard_output (subroutine), 1684
node_component_black_hole_standard_output_count (subroutine), 1684
node_component_black_hole_standard_output_merger (subroutine), 1684
node_component_black_hole_standard_output_properties (subroutine), 1684
node_component_black_hole_standard_post_evolve (subroutine), 1685
node_component_black_hole_standard_potential (function), 1686
node_component_black_hole_standard_radiative_efficiency (function), 1685
node_component_black_hole_standard_rate_compute (subroutine), 1685
node_component_black_hole_standard_rotation_curve (function), 1686
node_component_black_hole_standard_rotation_curve_gradient (function), 1686
node_component_black_hole_standard_satellite_merging (subroutine), 1685
node_component_black_hole_standard_scale_set (subroutine), 1685
node_component_black_hole_standard_seed_mass (function), 1484
node_component_black_hole_standard_seed_spin (function), 1484
node_component_black_hole_standard_spin (function), 1484
node_component_black_hole_standard_structure_tasks (module), 1686
node_component_black_hole_standard_triple_interaction (subroutine), 1685
node_component_black_hole_builder (subroutine), 1484
node_component_blackhole_destroy (subroutine), 1485
node_component_blackhole_dump (subroutine), 1485
node_component_blackhole_initializer (subroutine), 1485
node_component_blackhole_move (subroutine), 1485
node_component_blackhole_output (subroutine), 1485
node_component_blackhole_output_count (subroutine), 1485
node_component_blackhole_output_names (subroutine), 1485
node_component_blackhole_remove (subroutine), 1485
node_component_blackhole_type (function), 1485
node_component_blackholenull_builder (subroutine), 1486
node_component_blackholenull_count (function), 1486
node_component_blackholenull_deserialize_rates (subroutine), 1486
node_component_blackholenull_deserialize_scales (subroutine), 1486
node_component_blackholenull_deserialize_values (subroutine), 1486
node_component_blackholenull_destroy (subroutine), 1486
node_component_blackholenull_dump (subroutine), 1486
node_component_blackholenull_dump_raw (subroutine), 1486
node_component_blackholenull_initializer (subroutine), 1487
node_component_blackholenull_is_active (function), 1487
node_component_blackholenull_name_from_index (subroutine), 1487
node_component_blackholenull_ode_step_rates_init (subroutine), 1487
node_component_blackholenull_odestepsscalesinit (subroutine), 1487
node_component_blackholenull_output_count (subroutine), 1487
node_component_blackholenull_output_names (subroutine), 1487
node_component_blackholenull_read_raw (subroutine), 1487
node_component_blackholenullSerialize_rates (subroutine), 1487
CODE INDEX

node_component_blackholenull_serialize_scales (subroutine), 1488
node_component_blackholenull_serialize_values (subroutine), 1488
node_component_blackholenull_type (function), 1488
node_component_blackholesimple_builder (subroutine), 1488
node_component_blackholesimple_count (function), 1488
node_component_blackholesimple_deserialize_rates (subroutine), 1488
node_component_blackholesimple_deserialize_scales (subroutine), 1488
node_component_blackholesimple_deserialize_values (subroutine), 1488
node_component_blackholesimple_destroy (subroutine), 1488
node_component_blackholesimple_dump (subroutine), 1489
node_component_blackholesimple_dump_raw (subroutine), 1489
node_component_blackholesimple_dump_xml (subroutine), 1489
node_component_blackholesimple_initializor (subroutine), 1489
node_component_blackholesimple_is_active (function), 1489
node_component_blackholesimple_name_from_index (subroutine), 1489
node_component_blackholesimple_ode_step_rates_init (subroutine), 1489
node_component_blackholesimple_odestepscalesinit (subroutine), 1489
node_component_blackholesimple_output_count (subroutine), 1490
node_component_blackholesimple_output_names (subroutine), 1490
node_component_blackholesimple_read_raw (subroutine), 1490
node_component_blackholesimple_serialize_rates (subroutine), 1490
node_component_blackholesimple_serialize_scales (subroutine), 1490
node_component_blackholesimple_serialize_values (subroutine), 1490
node_component_blackholesimple_type (function), 1490
node_component_blackholestandard_builder (subroutine), 1490
node_component_blackholestandard_count (function), 1490
node_component_blackholestandard_deserialize_rates (subroutine), 1491
node_component_blackholestandard_deserialize_scales (subroutine), 1491
node_component_blackholestandard_deserialize_values (subroutine), 1491
node_component_blackholestandard_destroy (subroutine), 1491
node_component_blackholestandard_dump (subroutine), 1491
node_component_blackholestandard_dump_raw (subroutine), 1491
node_component_blackholestandard_dump_xml (subroutine), 1491
node_component_blackholestandard_initializor (subroutine), 1491
node_component_blackholestandard_is_active (function), 1491
node_component_blackholestandard_name_from_index (subroutine), 1492
node_component_blackholestandard_ode_step_rates_init (subroutine), 1492
node_component_blackholestandard_odestepscalesinit (subroutine), 1492
node_component_blackholestandard_output_count (subroutine), 1492
node_component_blackholestandard_output_names (subroutine), 1492
node_component_blackholestandard_read_raw (subroutine), 1492
node_component_blackholestandard_serialize_rates (subroutine), 1492
node_component_blackholestandard_serialize_scales (subroutine), 1492
node_component_blackholestandard_serialize_values (subroutine), 1492
node_component_blackholestandard_type (function), 1493
node_component_dark_matter_profile_scale (module), 1686
node_component_dark_matter_profile_scale_initialize (subroutine), 1687
node_component_dark_matter_profile_scale_initialize_scale (subroutine), 1687
node_component_dark_matter_profile_scale_plausibility (subroutine), 1687
node_component_dark_matter_profile_scale_preset (module), 1688
node_component_dark_matter_profile_scale_preset_promote (subroutine), 1688
node_component_dark_matter_profile_scale_preset_rate_compute (subroutine), 1688
node_component_dark_matter_profile_scale_preset_scale_set (subroutine), 1688
node_component_dark_matter_profile_scale_preset_tree_initialize (subroutine), 1688
node_component_dark_matter_profile_scale_promote (subroutine), 1687
node_component_dark_matter_profile_scale_rate_compute (subroutine), 1687
node_component_dark_matter_profile_scale_scale (function), 1687
node_component_dark_matter_profile_scale_scale_set (subroutine), 1687
node_component_dark_matter_profile_scale_shape (module), 1689
node_component_dark_matter_profile_scale_shape_initialize (subroutine), 1689
node_component_dark_matter_profile_scale_shape_initialize_shape (subroutine), 1689
node_component_dark_matter_profile_scale_shape_promote (subroutine), 1689
node_component_dark_matter_profile_scale_shape_rate_compute (subroutine), 1689
node_component_dark_matter_profile_scale_shape_scale_set (subroutine), 1689
node_component_dark_matter_profile_scale_shape_tree_initialize (subroutine), 1689
node_component_dark_matter_profile_scale_shape_tree_output (subroutine), 1689
node_component_dark_matter_profile_scale_tree_initialize (subroutine), 1687
node_component_dark_matter_profile_scale_tree_output (subroutine), 1688
node_component_darkmatterprofile_builder (subroutine), 1493
node_component_darkmatterprofile_destroy (subroutine), 1493
node_component_darkmatterprofile_dump (subroutine), 1493
node_component_darkmatterprofile_initializor (subroutine), 1493
node_component_darkmatterprofile_move (subroutine), 1493
node_component_darkmatterprofile_output (subroutine), 1493
node_component_darkmatterprofile_output_count (subroutine), 1493
node_component_darkmatterprofile_output_names (subroutine), 1493
node_component_darkmatterprofile_remove (subroutine), 1494
node_component_darkmatterprofile_type (function), 1494
node_component_darkmatterprofilenull_builder (subroutine), 1494
node_component_darkmatterprofilenull_count (function), 1494
node_component_darkmatterprofilenull_deserialize_rates (subroutine), 1494
node_component_darkmatterprofilenull_deserialize_scales (subroutine), 1494
node_component_darkmatterprofilenull_deserialize_values (subroutine), 1494
node_component_darkmatterprofilenull_destroy (subroutine), 1494
node_component_darkmatterprofilenull_dump (subroutine), 1494
node_component_darkmatterprofilenull_dump_raw (subroutine), 1495
node_component_darkmatterprofilenull_dump_xml (subroutine), 1495
node_component_darkmatterprofilenull_initializor (subroutine), 1495
node_component_darkmatterprofilenull_is_active (function), 1495
node_component_darkmatterprofilenull_name_from_index (subroutine), 1495
node_component_darkmatterprofilenull_ode_step_rates_init (subroutine), 1495
node_component_darkmatterprofilenull_odestepscalesinit (subroutine), 1495
node_component_darkmatterprofilenull_output_count (subroutine), 1495
node_component_darkmatterprofilenull_output_names (subroutine), 1496
node_component_darkmatterprofilenull_read_raw (subroutine), 1496
node_component_darkmatterprofilenull_serialize_rates (subroutine), 1496
node_component_darkmatterprofilenull_serialize_scales (subroutine), 1496
node_component_darkmatterprofilenull_serialize_values (subroutine), 1496
node_component_darkmatterprofilenull_type (function), 1496
node_component_darkmatterprofilescalescale_builder (subroutine), 1496
node_component_darkmatterprofilescalescale_count (function), 1496
node_component_darkmatterprofilescalescale_deserialize_rates (subroutine), 1496
node_component_darkmatterprofilescalescale_deserialize_scales (subroutine), 1497
node_component_darkmatterprofilescalescale_deserialize_values (subroutine), 1497
node_component_darkmatterprofilescalescale_destroy (subroutine), 1497
node_component_darkmatterprofilescalescale_dump (subroutine), 1497
node_component_darkmatterprofilescalescale_dump_raw (subroutine), 1497
node_component_darkmatterprofilescalescale_dump_xml (subroutine), 1497
node_component_darkmatterprofilescalescale_initializor (subroutine), 1497
node_component_darkmatterprofilescalescale_is_active (function), 1497
node_component_darkmatterprofilescalescale_name_from_index (subroutine), 1497
node_component_darkmatterprofilescalescale_ode_step_rates_init (subroutine), 1498
node_component_darkmatterprofilescalescale_output_count (subroutine), 1498
node_component_darkmatterprofilescalescale_output_names (subroutine), 1498
node_component_darkmatterprofilescalescale_read_raw (subroutine), 1498
node_component_darkmatterprofilescalescale_serialize_rates (subroutine), 1498
node_component_darkmatterprofilescalescale_serialize_scales (subroutine), 1498
node_component_darkmatterprofilescalescale_serialize_values (subroutine), 1498
node_component_darkmatterprofilescalescale_type (function), 1499
node_component_darkmatterprofilescalescalepreset_builder (subroutine), 1499
node_component_darkmatterprofilescalescalepreset_count (function), 1499
node_component_darkmatterprofilescalescalepreset_deserialize_rates (subroutine), 1499
node_component_darkmatterprofilescalescalepreset_deserialize_scales (subroutine), 1499
node_component_darkmatterprofilescalescalepreset_deserialize_values (subroutine), 1499
node_component_darkmatterprofilescalescalepreset_destroy (subroutine), 1499
node_component_darkmatterprofilescalescalepreset_dump (subroutine), 1499
node_component_darkmatterprofilescalescalepreset_dump_raw (subroutine), 1499
node_component_darkmatterprofilescalescalepreset_dump_xml (subroutine), 1500
node_component_darkmatterprofilescalescalepreset_initializor (subroutine), 1500
node_component_darkmatterprofilescalescalepreset_is_active (function), 1500
node_component_darkmatterprofilescalescalepreset_name_from_index (subroutine), 1500
node_component_darkmatterprofilescalescalepreset_ode_step_rates_init (subroutine), 1500
node_component_darkmatterprofilescalescalepreset_odestepscalesinit (subroutine), 1500
node_component_darkmatterprofilescalescalepreset_output_count (subroutine), 1500
node_component_darkmatterprofilescalescalepreset_output_names (subroutine), 1500
node_component_darkmatterprofilescalescalepreset_read_raw (subroutine), 1501
node_component_darkmatterprofilescalescalepreset_serialize_rates (subroutine), 1501
node_component_darkmatterprofilescalescalepreset_serialize_scales (subroutine), 1501
node_component_darkmatterprofilescalescalepreset_serialize_values (subroutine), 1501
node_component_darkmatterprofilescalescalepreset_type (function), 1501
node_component_darkmatterprofilescaleshapeshape_builder (subroutine), 1501
node_component_darkmatterprofilescaleshapeshape_count (function), 1501
node_component_darkmatterprofilescaleshapeshape_deserialize_rates (subroutine), 1501
node_component_darkmatterprofilescaleshapeshape_deserialize_scales (subroutine), 1501
node_component_darkmatterprofilescaleshapeshape_deserialize_values (subroutine), 1502
node_component_darkmatterprofilescaleshapeshape_destroy (subroutine), 1502
CODE INDEX

node_component_darkmatterprofilescaleshape_dump (subroutine), 1502
node_component_darkmatterprofilescaleshape_dump_raw (subroutine), 1502
node_component_darkmatterprofilescaleshape_dump_xml (subroutine), 1502
node_component_darkmatterprofilescaleshape_initializer (subroutine), 1502
node_component_darkmatterprofilescaleshape_is_active (function), 1502
node_component_darkmatterprofilescaleshape_name_from_index (subroutine), 1502
node_component_darkmatterprofilescaleshape_ode_step_rates_init (subroutine), 1503
node_component_darkmatterprofilescaleshape_odestepscalesinit (subroutine), 1503
node_component_darkmatterprofilescaleshape_output_count (subroutine), 1503
node_component_darkmatterprofilescaleshape_output_names (subroutine), 1503
node_component_darkmatterprofilescaleshape_read_raw (subroutine), 1503
node_component_darkmatterprofilescaleshape_serialize_rates (subroutine), 1503
node_component_darkmatterprofilescaleshape_serialize_scales (subroutine), 1503
node_component_darkmatterprofilescaleshape_serialize_values (subroutine), 1503
node_component_density_null (function), 1504
node_component_deserialize_null (subroutine), 1504
node_component_disk_builder (subroutine), 1504
node_component_disk_destroy (subroutine), 1504
node_component_disk_dump (subroutine), 1504
node_component_disk_exponential (module), 1690
node_component_disk_exponential_attach_pipes (subroutine), 1504
node_component_disk_exponential_calculation_reset (subroutine), 1690
node_component_disk_exponential_create (subroutine), 1690
node_component_disk_exponential_data (module), 1692
node_component_disk_exponential_density (function), 1504
node_component_disk_exponential_enclosed_mass (function), 1504
node_component_disk_exponential_enclosed_mass_dimensionless (function), 1693
node_component_disk_exponential_half_mass_radius (function), 1505
node_component_disk_exponential_initialize (subroutine), 1690
node_component_disk_exponential_post_evolve (subroutine), 1690
node_component_disk_exponential_potential (function), 1505
node_component_disk_exponential_pre_evolve (subroutine), 1691
node_component_disk_exponential_radius_solve (function), 1691
node_component_disk_exponential_radius_solve_set (subroutine), 1691
node_component_disk_exponential_radius_solver (subroutine), 1691
node_component_disk_exponential_radius_solver_plausibility (subroutine), 1691
node_component_disk_exponential_rate_compute (subroutine), 1691
node_component_disk_exponential_reset (subroutine), 1693
node_component_disk_exponential_rotation_curve (function), 1505
node_component_disk_exponential_rotation_curve_bessel_factors (function), 1693
node_component_disk_exponential_rotation_curve_gradient (function), 1505
node_component_disk_exponential_rttn_crv_grdnt_bssl_fctrs (function), 1693
node_component_disk_exponential_satellite_merging (subroutine), 1692
node_component_disk_exponential_scale_set (subroutine), 1692
node_component_disk_exponential_star_formation_history_output (subroutine), 1692
node_component_disk_exponential_star_formation_rate (function), 1692
node_component_disk_exponential_state_retrieve (subroutine), 1693
node_component_disk_exponential_state_store (subroutine), 1693

2132
node_component_disk_exponential_surface_density (function), 1505
node_component_disk_exponential_velocity (function), 1692
node_component_disk_exponential_velocity_set (subroutine), 1692
node_component_disk_initializor (subroutine), 1505
node_component_disk_move (subroutine), 1505
node_component_disk_output (subroutine), 1506
node_component_disk_output_count (subroutine), 1506
node_component_disk_output_names (subroutine), 1506
node_component_disk_remove (subroutine), 1506
node_component_disk_type (function), 1506
node_component_disk_very_simple (module), 1694
node_component_disk_very_simple_attach_pipe (subroutine), 1506
node_component_disk_very_simple_enclosed_mass (function), 1506
node_component_disk_very_simple_initialize (subroutine), 1694
node_component_disk_very_simple_post_evolve (subroutine), 1694
node_component_disk_very_simple_rate_compute (subroutine), 1694
node_component_disk_very_simple_satellite_merging (subroutine), 1694
node_component_disk_very_simple_scale_set (subroutine), 1694
node_component_disk_very_simple_sfr (function), 1694
node_component_diskexponential_builder (subroutine), 1506
node_component_diskexponential_count (function), 1506
node_component_diskexponential_deserialize_rates (subroutine), 1507
node_component_diskexponential_deserialize_scales (subroutine), 1507
node_component_diskexponential_deserialize_values (subroutine), 1507
node_component_diskexponential_destroy (subroutine), 1507
node_component_diskexponential_dump (subroutine), 1507
node_component_diskexponential_dump_raw (subroutine), 1507
node_component_diskexponential_dump_xml (subroutine), 1507
node_component_diskexponential_initializor (subroutine), 1507
node_component_diskexponential_is_active (function), 1507
node_component_diskexponential_name_from_index (subroutine), 1508
node_component_diskexponential_ode_step_rates_init (subroutine), 1508
node_component_diskexponential_odestepscalesinit (subroutine), 1508
node_component_diskexponential_output_count (subroutine), 1508
node_component_diskexponential_output_names (subroutine), 1508
node_component_diskexponential_read_raw (subroutine), 1508
node_component_diskexponential_serialize_rates (subroutine), 1508
node_component_diskexponential_serialize_scales (subroutine), 1508
node_component_diskexponential_serialize_values (subroutine), 1508
node_component_diskexponential_serializze_scales (subroutine), 1508
node_component_disknull_builder (subroutine), 1509
node_component_disknull_count (function), 1509
node_component_disknull_deserialize_rates (subroutine), 1509
node_component_disknull_deserialize_scales (subroutine), 1509
node_component_disknull_deserialize_values (subroutine), 1509
node_component_disknull_destroy (subroutine), 1509
node_component_disknull_dump (subroutine), 1509
node_component_disknull_dump_raw (subroutine), 1510
node_component_disknull_dump_xml (subroutine), 1510
node_component_disknull_initialize (subroutine), 1510
node_component_disknull_is_active (function), 1510
node_component_disknull_name_from_index (subroutine), 1510
node_component_disknull_ode_step_rates_init (subroutine), 1510
node_component_disknull_odestepscalesinit (subroutine), 1510
node_component_disknull_output_count (subroutine), 1510
node_component_disknull_output_names (subroutine), 1510
node_component_disknull_read_raw (subroutine), 1511
node_component_disknull_serialize_rates (subroutine), 1511
node_component_disknull_serialize_scales (subroutine), 1511
node_component_disknull_serialize_values (subroutine), 1511
node_component_disknull_type (function), 1511
node_component_diskverysimple_builder (subroutine), 1511
node_component_diskverysimple_count (function), 1511
node_component_diskverysimple_deserialize_rates (subroutine), 1511
node_component_diskverysimple_deserialize_scales (subroutine), 1511
node_component_diskverysimple_deserialize_values (subroutine), 1512
node_component_diskverysimple_destroy (subroutine), 1512
node_component_diskverysimple_dump (subroutine), 1512
node_component_diskverysimple_dump_raw (subroutine), 1512
node_component_diskverysimple_dump_xml (subroutine), 1512
node_component_diskverysimple_initialize (subroutine), 1512
node_component_diskverysimple_is_active (function), 1512
node_component_diskverysimple_name_from_index (subroutine), 1512
node_component_diskverysimple_ode_step_rates_init (subroutine), 1512
node_component_diskverysimple_odestepscalesinit (subroutine), 1513
node_component_diskverysimple_output_count (subroutine), 1513
node_component_diskverysimple_output_names (subroutine), 1513
node_component_diskverysimple_read_raw (subroutine), 1513
node_component_diskverysimple_serialize_rates (subroutine), 1513
node_component_diskverysimple_serialize_scales (subroutine), 1513
node_component_diskverysimple_serialize_values (subroutine), 1513
node_component_diskverysimple_type (function), 1513
node_component_dump_null (subroutine), 1513
node_component_dump_raw_null (subroutine), 1514
node_component_dump_xml_null (subroutine), 1514
node_component_dynamics_statistics_bars (module), 1695
node_component_dynamics_statistics_bars_output (subroutine), 1695
node_component_dynamics_statistics_bars_record (subroutine), 1514, 1695
node_component_dynamics_statistics_bars_timestep (subroutine), 1695
node_component_dynamicsstatistics_builder (subroutine), 1514
node_component_dynamicsstatistics_destroy (subroutine), 1514
node_component_dynamicsstatistics_dump (subroutine), 1514
node_component_dynamicsstatistics_initialize (subroutine), 1514
node_component_dynamicsstatistics_move (subroutine), 1514
node_component_dynamicsstatistics_output (subroutine), 1514
node_component_dynamicsstatistics_output_count (subroutine), 1515
node_component_dynamicsstatistics_output_names (subroutine), 1515
node_component_dynamicsstatistics_remove (subroutine), 1515
node_component_dynamicsstatistics_type (function), 1515
node_component_dynamicsstatisticsbars_builder (subroutine), 1515
node_component_dynamicsstatisticsbars_count (function), 1515
node_component_dynamicsstatisticsbars_deserialize_rates (subroutine), 1515
node_component_dynamicsstatisticsbars_deserialize_scales (subroutine), 1515
node_component_dynamicsstatisticsbars_deserialize_values (subroutine), 1515
node_component_dynamicsstatisticsbars_destroy (subroutine), 1516
node_component_dynamicsstatisticsbars_dump (subroutine), 1516
node_component_dynamicsstatisticsbars_dump_raw (subroutine), 1516
node_component_dynamicsstatisticsbars_dump_xml (subroutine), 1516
node_component_dynamicsstatisticsbars_initializor (subroutine), 1516
node_component_dynamicsstatisticsbars_is_active (function), 1516
node_component_dynamicsstatisticsbars_name_from_index (subroutine), 1516
node_component_dynamicsstatisticsbars_ode_step_rates_init (subroutine), 1516
node_component_dynamicsstatisticsbars_output_count (subroutine), 1517
node_component_dynamicsstatisticsbars_output_names (subroutine), 1517
node_component_dynamicsstatisticsbars_read_raw (subroutine), 1517
node_component_dynamicsstatisticsbars_serialize_rates (subroutine), 1517
node_component_dynamicsstatisticsbars_serialize_scales (subroutine), 1517
node_component_dynamicsstatisticsbars_serialize_values (subroutine), 1517
node_component_dynamicsstatisticsbars_type (function), 1517
node_component_dynamicsstatisticsnull_builder (subroutine), 1517
node_component_dynamicsstatisticsnull_count (function), 1518
node_component_dynamicsstatisticsnull_deserialize_rates (subroutine), 1518
node_component_dynamicsstatisticsnull_deserialize_scales (subroutine), 1518
node_component_dynamicsstatisticsnull_deserialize_values (subroutine), 1518
node_component_dynamicsstatisticsnull_destroy (subroutine), 1518
node_component_dynamicsstatisticsnull_dump (subroutine), 1518
node_component_dynamicsstatisticsnull_dump_raw (subroutine), 1518
node_component_dynamicsstatisticsnull_dump_xml (subroutine), 1518
node_component_dynamicsstatisticsnull_initializor (subroutine), 1518
node_component_dynamicsstatisticsnull_is_active (function), 1519
node_component_dynamicsstatisticsnull_name_from_index (subroutine), 1519
node_component_dynamicsstatisticsnull_ode_step_rates_init (subroutine), 1519
node_component_dynamicsstatisticsnull_odestepsscalesinit (subroutine), 1519
node_component_dynamicsstatisticsnull_output_count (subroutine), 1519
node_component_dynamicsstatisticsnull_output_names (subroutine), 1519
node_component_dynamicsstatisticsnull_read_raw (subroutine), 1519
node_component_dynamicsstatisticsnull_serialize_rates (subroutine), 1519
node_component_dynamicsstatisticsnull_serialize_scales (subroutine), 1520
node_component_dynamicsstatisticsnull_serialize_values (subroutine), 1520
node_component_dynamicsstatisticsnull_type (function), 1520
node_component_enclosed_mass_null (function), 1520
node_component_formation_time_cole2000_create (subroutine), 1696
node_component_formation_time_cole2000_node_promotion (subroutine), 1696
node_component_formation_time_cole2000_rate_compute (subroutine), 1696
node_component_formation_time_cole2000_tree_initialize (subroutine), 1696
node_component_formation_times_cole2000 (module), 1695
node_component_formation_times_cole2000_initialize (subroutine), 1696
node_component_formationtime_builder (subroutine), 1520
node_component_formationtime_destroy (subroutine), 1520
node_component_formationtime_dump (subroutine), 1520
node_component_formationtime_initializer (subroutine), 1520
node_component_formationtime_move (subroutine), 1520
node_component_formationtime_output (subroutine), 1521
node_component_formationtime_output_count (subroutine), 1521
node_component_formationtime_output_names (subroutine), 1521
node_component_formationtime_remove (subroutine), 1521
node_component_formationtime_type (function), 1521
node_component_formationtimecole2000_builder (subroutine), 1521
node_component_formationtimecole2000_count (function), 1521
node_component_formationtimecole2000_deserialize_rates (subroutine), 1521
node_component_formationtimecole2000_deserialize_scales (subroutine), 1521
node_component_formationtimecole2000_deserialize_values (subroutine), 1521
node_component_formationtimecole2000_destroy (subroutine), 1522
node_component_formationtimecole2000_dump (subroutine), 1522
node_component_formationtimecole2000_dump_raw (subroutine), 1522
node_component_formationtimecole2000_dump_xml (subroutine), 1522
node_component_formationtimecole2000_initializor (subroutine), 1522
node_component_formationtimecole2000_is_active (function), 1522
node_component_formationtimecole2000_name_from_index (subroutine), 1522
node_component_formationtimecole2000_ode_step_rates_init (subroutine), 1523
node_component_formationtimecole2000_odestepscalesinit (subroutine), 1523
node_component_formationtimecole2000_output_count (subroutine), 1523
node_component_formationtimecole2000_output_names (subroutine), 1523
node_component_formationtimecole2000_read_raw (subroutine), 1523
node_component_formationtimecole2000_serialize_rates (subroutine), 1523
node_component_formationtimecole2000_serialize_scales (subroutine), 1523
node_component_formationtimecole2000_serialize_values (subroutine), 1523
node_component_formationtimecole2000_type (function), 1523
node_component_formationtimenull_builder (subroutine), 1524
node_component_formationtimenull_count (function), 1524
node_component_formationtimenull_deserialize_rates (subroutine), 1524
node_component_formationtimenull_deserialize_scales (subroutine), 1524
node_component_formationtimenull_deserialize_values (subroutine), 1524
node_component_formationtimenull_destroy (subroutine), 1524
node_component_formationtimenull_dump (subroutine), 1524
node_component_formationtimenull_dump_raw (subroutine), 1524
node_component_formationtimenull_dump_xml (subroutine), 1524
node_component_formationtimenull_initializer (subroutine), 1525
node_component_formationtimenull_is_active (function), 1525
node_component_formationtimenull_name_from_index (subroutine), 1525
node_component_formationtimenull_ode_step_rates_init (subroutine), 1525
node_component_formationtimenull_odestepscalesinit (subroutine), 1525
node_component_formationtimenull_output_count (subroutine), 1525
node_component_formationtimenull_output_names (subroutine), 1525
node_component_formationtimenull_read_raw (subroutine), 1525
node_component_formationtimenull_serialize_rates (subroutine), 1526
node_component_formationtimenull_serialize_scales (subroutine), 1526
node_component_formationtimenull_serialize_values (subroutine), 1526
node_component_formationtimenull_type (function), 1526
node_component_generic_destroy (subroutine), 1526
node_component_generic_type (function), 1526
node_component_host_history_standard (module), 1696
node_component_host_history_standard_merger_tree_init (subroutine), 1696
node_component_host_history_standard_update_history (subroutine), 1697
node_component_host_node (function), 1526
node_component_hosthistory_builder (subroutine), 1526
node_component_hosthistory_destroy (subroutine), 1526
node_component_hosthistory_dump (subroutine), 1526
node_component_hosthistory_initializer (subroutine), 1527
node_component_hosthistory_move (subroutine), 1527
node_component_hosthistory_output (subroutine), 1527
node_component_hosthistory_output_count (subroutine), 1527
node_component_hosthistory_output_names (subroutine), 1527
node_component_hosthistory_remove (subroutine), 1527
node_component_hosthistory_type (function), 1527
node_component_hosthistorynull_builder (subroutine), 1527
node_component_hosthistorynull_count (function), 1528
node_component_hosthistorynull_deserialize_rates (subroutine), 1528
node_component_hosthistorynull_deserialize_scales (subroutine), 1528
node_component_hosthistorynull_deserialize_values (subroutine), 1528
node_component_hosthistorynull_destroy (subroutine), 1528
node_component_hosthistorynull_dump (subroutine), 1528
node_component_hosthistorynull_dump_raw (subroutine), 1528
node_component_hosthistorynull_dump_xml (subroutine), 1528
node_component_hosthistorynull_initializor (subroutine), 1528
node_component_hosthistorynull_is_active (function), 1529
node_component_hosthistorynull_name_from_index (subroutine), 1529
node_component_hosthistorynull_ode_step_rates_init (subroutine), 1529
node_component_hosthistorynull_odestep_scalesinit (subroutine), 1529
node_component_hosthistorynull_output_count (subroutine), 1529
node_component_hosthistorynull_output_names (subroutine), 1529
node_component_hosthistorynull_read_raw (subroutine), 1529
node_component_hosthistorynull_serialize_rates (subroutine), 1529
node_component_hosthistorynull_serialize_scales (subroutine), 1529
node_component_hosthistorynull_serialize_values (subroutine), 1530
node_component_hosthistorynull_type (function), 1530
node_component_hosthistorystandard_builder (subroutine), 1530
node_component_hosthistorystandard_count (function), 1530
node_component_hosthistorystandard_deserialize_rates (subroutine), 1530
node_component_hosthistorystandard_deserialize_scales (subroutine), 1530
node_component_hosthistorystandard_deserialize_values (subroutine), 1530
node_component_hosthistorystandard_destroy (subroutine), 1530
node_component_hosthistorystandard_dump (subroutine), 1530
node_component_hosthistorystandard_dump_raw (subroutine), 1531
node_component_hosthistorystandard_dump_xml (subroutine), 1531
node_component_hosthistorystandard_initializor (subroutine), 1531
node_component_hosthistorystandard_is_active (function), 1531
node_component_hosthistorystandard_name_from_index (subroutine), 1531
node_component_hosthistorystandard_ode_step_rates_init (subroutine), 1531
node_component_hosthistorystandard_output_count (subroutine), 1531
node_component_hosthistorystandard_output_names (subroutine), 1532
node_component_hosthistorystandard_output_names (subroutine), 1532
node_component_hosthistorystandard_serialize_rates (subroutine), 1532
node_component_hosthistorystandard_serialize_scales (subroutine), 1532
node_component_hosthistorystandard_serialize_values (subroutine), 1532
node_component_hosthistorystandard_type (function), 1532
node_component_hot_halo_cold_mode (module), 1697
node_component_hot_halo_cold_mode_dense_task (function), 1699
node_component_hot_halo_cold_mode_enclosed_mass_task (function), 1699
node_component_hot_halo_cold_mode_formation (subroutine), 1697
node_component_hot_halo_cold_mode_initialize (subroutine), 1697
node_component_hot_halo_cold_mode_node_merger (subroutine), 1697
node_component_hot_halo_cold_mode_outflow_return (subroutine), 1697
node_component_hot_halo_cold_mode_promote (subroutine), 1698
node_component_hot_halo_cold_mode_push_to_cooling_pipes (subroutine), 1698
node_component_hot_halo_cold_mode_rate_compute (subroutine), 1698
node_component_hot_halo_cold_mode_rotation_curve_gradient_task (function), 1699
node_component_hot_halo_cold_mode_rotation_curve_task (function), 1700
node_component_hot_halo_cold_mode_satellite_merger (subroutine), 1698
node_component_hot_halo_cold_mode_scale_set (subroutine), 1698
node_component_hot_halo_cold_mode_structure_tasks (module), 1699
node_component_hot_halo_cold_mode_thread_initialize (subroutine), 1698
node_component_hot_halo_cold_mode_tree_initialize (subroutine), 1699
node_component_hot_halo_outflow_tracking (module), 1700
node_component_hot_halo_outflow_tracking_rate_compute (subroutine), 1700
node_component_hot_halo_outflow_tracking_scale_set (subroutine), 1700
node_component_hot_halo_standard (module), 1700
node_component_hot_halo_standard_cooling_rate (subroutine), 1701
node_component_hot_halo_standard_create (subroutine), 1701
node_component_hot_halo_standard_data (module), 1705
node_component_hot_halo_standard_formation (subroutine), 1701
node_component_hot_halo_standard_heat_source (subroutine), 1701
node_component_hot_halo_standard_hot_gas_all_rate (subroutine), 1701
node_component_hot_halo_standard_initialize (subroutine), 1701
node_component_hot_halo_standard_initializor (subroutine), 1701
node_component_hot_halo_standard_mass_removal_rate (subroutine), 1701
node_component_hot_halo_standard_mass_sink (subroutine), 1702
node_component_hot_halo_standard_node_merger (subroutine), 1702
node_component_hot_halo_standard_outer_radius (function), 1702
node_component_hot_halo_standard_outer_radius_growth_rate (function), 1702
node_component_hot_halo_standard_outflow_return (subroutine), 1702
node_component_hot_halo_standard_outflow_stripped_fraction (function), 1702
node_component_hot_halo_standard_outflowing_abundances_rate (subroutine), 1703
node_component_hot_halo_standard_outflowing_ang_mom_rate (subroutine), 1703
node_component_hot_halo_standard_outflowing_mass_rate (subroutine), 1703
node_component_hot_halo_standard_post_evolve (subroutine), 1703
node_component_hot_halo_standard_pre_evolve (subroutine), 1703
node_component_hot_halo_standard_promote (subroutine), 1703
node_component_hot_halo_standard_push_from_halo (subroutine), 1703
node_component_hot_halo_standard_push_to_cooling_pipes (subroutine), 1703
node_component_hot_halo_standard_rate_compute (subroutine), 1704
node_component_hot_halo_standard_reset (subroutine), 1704
node_component_hot_halo_standard_satellite_merger (subroutine), 1704
node_component_hot_halo_standard_scale_set (subroutine), 1704
node_component_hot_halo_standard_strip_gas_rate (subroutine), 1704
node_component_hot_halo_standard_thread_initialize (subroutine), 1704
node_component_hot_halo_standard_tree_initialize (subroutine), 1704
node_component_hot_halo_very_simple (module), 1705
node_component_hot_halo_very_simple_cooling_rate (subroutine), 1706
node_component_hot_halo_very_simple_create (subroutine), 1706
node_component_hot_halo_very_simple_initialize (subroutine), 1706
node_component_hot_halo_very_simple_node_merger (subroutine), 1706
node_component_hot_halo_very_simple_outer_radius (function), 1706
node_component_hot_halo_very_simple_outflowing_mass_rate (subroutine), 1706
node_component_hot_halo_very_simple_post_evolve (subroutine), 1706
node_component_hot_halo_very_simple_promote (subroutine), 1706
node_component_hot_halo_very_simple_push_to_cooling_pipes (subroutine), 1706
node_component_hot_halo_very_simple_rate_compute (subroutine), 1707
node_component_hot_halo_very_simple_reset (subroutine), 1707
node_component_hot_halo_very_simple_satellite_merger (subroutine), 1707
node_component_hot_halo_very_simple_scale_set (subroutine), 1707
node_component_hot_halo_very_simple_tree_initialize (subroutine), 1707
node_component_hothalo_builder (subroutine), 1532
node_component_hothalo_destroy (subroutine), 1532
node_component_hothalo_dump (subroutine), 1533
node_component_hothalo_initializor (subroutine), 1533
node_component_hothalo_move (subroutine), 1533
node_component_hothalo_output (subroutine), 1533
node_component_hothalo_output_count (subroutine), 1533
node_component_hothalo_output_names (subroutine), 1533
node_component_hothalo_remove (subroutine), 1533
node_component_hothalo_type (function), 1533
node_component_hothalocoldmode_builder (subroutine), 1533
node_component_hothalocoldmode_count (function), 1534
node_component_hothalocoldmode_deserialize_rates (subroutine), 1534
node_component_hothalocoldmode_deserialize_scales (subroutine), 1534
node_component_hothalocoldmode_deserialize_values (subroutine), 1534
node_component_hothalocoldmode_destroy (subroutine), 1534
node_component_hothalocoldmode_dump (subroutine), 1534
node_component_hothalocoldmode_dump_raw (subroutine), 1534
node_component_hothalocoldmode_dump_xml (subroutine), 1534
CODE INDEX

node_component_hothalocoldmode_initializor (subroutine), 1535
node_component_hothalocoldmode_is_active (function), 1535
node_component_hothalocoldmode_name_from_index (subroutine), 1535
node_component_hothalocoldmode_ode_step_rates_init (subroutine), 1535
node_component_hothalocoldmode_odestepscalesinit (subroutine), 1535
node_component_hothalocoldmode_output_count (subroutine), 1535
node_component_hothalocoldmode_output_names (subroutine), 1535
node_component_hothalocoldmode_read_raw (subroutine), 1535
node_component_hothalocoldmode_serialize_rates (subroutine), 1535
node_component_hothalocoldmode_serialize_scales (subroutine), 1536
node_component_hothalocoldmode_serialize_values (subroutine), 1536
node_component_hothalocoldmode_type (function), 1536
node_component_hothalonull_builder (subroutine), 1536
node_component_hothalonull_count (function), 1536
node_component_hothalonull_deserialize_rates (subroutine), 1536
node_component_hothalonull_deserialize_scales (subroutine), 1536
node_component_hothalonull_deserialize_values (subroutine), 1536
node_component_hothalonull_destroy (subroutine), 1536
node_component_hothalonull_dump (subroutine), 1537
node_component_hothalonull_dump_raw (subroutine), 1537
node_component_hothalonull_dump_xml (subroutine), 1537
node_component_hothalonull_initializor (subroutine), 1537
node_component_hothalonull_is_active (function), 1537
node_component_hothalonull_name_from_index (subroutine), 1537
node_component_hothalonull_ode_step_rates_init (subroutine), 1537
node_component_hothalonull_odestepscalesinit (subroutine), 1537
node_component_hothalonull_output_count (subroutine), 1537
node_component_hothalonull_output_names (subroutine), 1538
node_component_hothalonull_read_raw (subroutine), 1538
node_component_hothalonull_serialize_rates (subroutine), 1538
node_component_hothalonull_serialize_scales (subroutine), 1538
node_component_hothalonull_serialize_values (subroutine), 1538
node_component_hothalonull_type (function), 1538
node_component_hothalouflowtracking_builder (subroutine), 1538
node_component_hothalouflowtracking_count (function), 1538
node_component_hothalouflowtracking_deserialize_rates (subroutine), 1538
node_component_hothalouflowtracking_deserialize_scales (subroutine), 1539
node_component_hothalouflowtracking_deserialize_values (subroutine), 1539
node_component_hothalouflowtracking_destroy (subroutine), 1539
node_component_hothalouflowtracking_dump (subroutine), 1539
node_component_hothalouflowtracking_dump_raw (subroutine), 1539
node_component_hothalouflowtracking_dump_xml (subroutine), 1539
node_component_hothalouflowtracking_initializor (subroutine), 1539
node_component_hothalouflowtracking_is_active (function), 1539
node_component_hothalouflowtracking_name_from_index (subroutine), 1539
node_component_hothalouflowtracking_ode_step_rates_init (subroutine), 1540
node_component_hothalouflowtracking_odestepscalesinit (subroutine), 1540
node_component_hothalouflowtracking_output_count (subroutine), 1540
node_component_hothalouflowtracking_output_names (subroutine), 1540

2140
<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Line Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_component_hothalooutflowtracking_read_raw</td>
<td>1540</td>
</tr>
<tr>
<td>node_component_hothalooutflowtracking_serialize_rates</td>
<td>1540</td>
</tr>
<tr>
<td>node_component_hothalooutflowtracking_serialize_scales</td>
<td>1540</td>
</tr>
<tr>
<td>node_component_hothalooutflowtracking_serialize_values</td>
<td>1540</td>
</tr>
<tr>
<td>node_component_hothalooutflowtracking_type</td>
<td>1541</td>
</tr>
<tr>
<td>node_component_hothalostandard_builder</td>
<td>1541</td>
</tr>
<tr>
<td>node_component_hothalostandard_count</td>
<td>1541</td>
</tr>
<tr>
<td>node_component_hothalostandard_deserialize_rates</td>
<td>1541</td>
</tr>
<tr>
<td>node_component_hothalostandard_deserialize_scales</td>
<td>1541</td>
</tr>
<tr>
<td>node_component_hothalostandard_deserialize_values</td>
<td>1541</td>
</tr>
<tr>
<td>node_component_hothalostandard_destroy</td>
<td>1541</td>
</tr>
<tr>
<td>node_component_hothalostandard_dump</td>
<td>1541</td>
</tr>
<tr>
<td>node_component_hothalostandard_dump_raw</td>
<td>1541</td>
</tr>
<tr>
<td>node_component_hothalostandard_dump_xml</td>
<td>1542</td>
</tr>
<tr>
<td>node_component_hothalostandard_initializor</td>
<td>1542</td>
</tr>
<tr>
<td>node_component_hothalostandard_is_active</td>
<td>1542</td>
</tr>
<tr>
<td>node_component_hothalostandard_name_from_index</td>
<td>1542</td>
</tr>
<tr>
<td>node_component_hothalostandard_ode_step_rates_init</td>
<td>1542</td>
</tr>
<tr>
<td>node_component_hothalostandard_odestepscalesinit</td>
<td>1542</td>
</tr>
<tr>
<td>node_component_hothalostandard_output_count</td>
<td>1542</td>
</tr>
<tr>
<td>node_component_hothalostandard_output_names</td>
<td>1542</td>
</tr>
<tr>
<td>node_component_hothalostandard_read_raw</td>
<td>1542</td>
</tr>
<tr>
<td>node_component_hothalostandard_serialize_rates</td>
<td>1543</td>
</tr>
<tr>
<td>node_component_hothalostandard_serialize_scales</td>
<td>1543</td>
</tr>
<tr>
<td>node_component_hothalostandard_serialize_values</td>
<td>1543</td>
</tr>
<tr>
<td>node_component_hothalostandard_type</td>
<td>1543</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_builder</td>
<td>1543</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_count</td>
<td>1543</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_deserialize_rates</td>
<td>1543</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_deserialize_scales</td>
<td>1543</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_deserialize_values</td>
<td>1543</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_destroy</td>
<td>1544</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_dump</td>
<td>1544</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_dump_raw</td>
<td>1544</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_dump_xml</td>
<td>1544</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_initializor</td>
<td>1544</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_is_active</td>
<td>1544</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_name_from_index</td>
<td>1544</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_ode_step_rates_init</td>
<td>1544</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_odestepscalesinit</td>
<td>1545</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_output_count</td>
<td>1545</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_output_names</td>
<td>1545</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_read_raw</td>
<td>1545</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_serialize_rates</td>
<td>1545</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_serialize_scales</td>
<td>1545</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_serialize_values</td>
<td>1545</td>
</tr>
<tr>
<td>node_component_hothaloverysimple_type</td>
<td>1545</td>
</tr>
<tr>
<td>node_component_indices_builder</td>
<td>1545</td>
</tr>
<tr>
<td>node_component_indices_destroy</td>
<td>1546</td>
</tr>
</tbody>
</table>
node_component_indices_dump (subroutine), 1546
node_component_indices_initializor (subroutine), 1546
node_component_indices_move (subroutine), 1546
node_component_indices_output (subroutine), 1546
node_component_indices_output_count (subroutine), 1546
node_component_indices_output_names (subroutine), 1546
node_component_indices_remove (subroutine), 1546
node_component_indices_standard (module), 1707
node_component_indices_standard_merger_tree_init (subroutine), 1707
node_component_indices_type (function), 1546
node_component_indicesnull_builder (subroutine), 1547
node_component_indicesnull_count (function), 1547
node_component_indicesnull_deserialize_rates (subroutine), 1547
node_component_indicesnull_deserialize_scales (subroutine), 1547
node_component_indicesnull_deserialize_values (subroutine), 1547
node_component_indicesnull_destroy (subroutine), 1547
node_component_indicesnull_dump (subroutine), 1547
node_component_indicesnull_dump_raw (subroutine), 1547
node_component_indicesnull_dump_xml (subroutine), 1547
node_component_indicesnull_initializor (subroutine), 1548
node_component_indicesnull_is_active (function), 1548
node_component_indicesnull_name_from_index (subroutine), 1548
node_component_indicesnull_ode_step_rates_init (subroutine), 1548
node_component_indicesnull_odestepscalesinit (subroutine), 1548
node_component_indicesnull_output_count (subroutine), 1548
node_component_indicesnull_output_names (subroutine), 1548
node_component_indicesnull_read_raw (subroutine), 1548
node_component_indicesnull_serialize_rates (subroutine), 1548
node_component_indicesnull_serialize_scales (subroutine), 1549
node_component_indicesnull_serialize_values (subroutine), 1549
node_component_indicesnull_type (function), 1549
node_component_indicesstandard_builder (subroutine), 1549
node_component_indicesstandard_count (function), 1549
node_component_indicesstandard_deserialize_rates (subroutine), 1549
node_component_indicesstandard_deserialize_scales (subroutine), 1549
node_component_indicesstandard_deserialize_values (subroutine), 1549
node_component_indicesstandard_destroy (subroutine), 1549
node_component_indicesstandard_dump (subroutine), 1550
node_component_indicesstandard_dump_raw (subroutine), 1550
node_component_indicesstandard_dump_xml (subroutine), 1550
node_component_indicesstandard_initializor (subroutine), 1550
node_component_indicesstandard_is_active (function), 1550
node_component_indicesstandard_name_from_index (subroutine), 1550
node_component_indicesstandard_ode_step_rates_init (subroutine), 1550
node_component_indicesstandard_odestepscalesinit (subroutine), 1550
node_component_indicesstandard_output_count (subroutine), 1550
node_component_indicesstandard_output_names (subroutine), 1551
node_component_indicesstandard_read_raw (subroutine), 1551
node_component_indicesstandard_serialize_rates (subroutine), 1551
node_component_indicesstandard_serialize_scales (subroutine), 1551
node_component_indicesstandard_serialize_values (subroutine), 1551
node_component_indicesstandard_type (function), 1551
node_component_inter_output_standard (module), 1707
node_component_inter_output_standard_rate_compute (subroutine), 1708
node_component_inter_output_standard_reset (subroutine), 1708
node_component_inter_output_standard_satellite_merge (subroutine), 1708
node_component_inter_output_standard_scale_set (subroutine), 1708
node_component_inter_output_builder (subroutine), 1551
node_component_inter_output_destroy (subroutine), 1551
node_component_interoutput_dump (subroutine), 1551
node_component_interoutput_initializor (subroutine), 1552
node_component_interoutput_move (subroutine), 1552
node_component_interoutput_output (subroutine), 1552
node_component_interoutput_output_count (subroutine), 1552
node_component_interoutput_output_names (subroutine), 1552
node_component_interoutput_remove (subroutine), 1552
node_component_interoutput_type (function), 1552
node_component_interoutputnull_builder (subroutine), 1552
node_component_interoutputnull_count (function), 1553
node_component_interoutputnull_deserialize_rates (subroutine), 1553
node_component_interoutputnull_deserialize_scales (subroutine), 1553
node_component_interoutputnull_deserialize_values (subroutine), 1553
node_component_interoutputnull_destroy (subroutine), 1553
node_component_interoutputnull_dump (subroutine), 1553
node_component_interoutputnull_dump_raw (subroutine), 1553
node_component_interoutputnull_dump_xml (subroutine), 1553
node_component_interoutputnull_initializor (subroutine), 1553
node_component_interoutputnull_is_active (function), 1554
node_component_interoutputnull_name_from_index (subroutine), 1554
node_component_interoutputnull_ode_step_rates_init (subroutine), 1554
node_component_interoutputnull_odestep_scales_init (subroutine), 1554
node_component_interoutputnull_output_count (subroutine), 1554
node_component_interoutputnull_output_names (subroutine), 1554
node_component_interoutputnull_read_raw (subroutine), 1554
node_component_interoutputnull_serialize_rates (subroutine), 1554
node_component_interoutputnull_serialize_scales (subroutine), 1554
node_component_interoutputnull_serialize_values (subroutine), 1555
node_component_interoutputnull_type (function), 1555
node_component_interoutputstandard_builder (subroutine), 1555
node_component_interoutputstandard_count (function), 1555
node_component_interoutputstandard_deserialize_rates (subroutine), 1555
node_component_interoutputstandard_deserialize_scales (subroutine), 1555
node_component_interoutputstandard_deserialize_values (subroutine), 1555
node_component_interoutputstandard_destroy (subroutine), 1555
node_component_interoutputstandard_dump (subroutine), 1555
node_component_interoutputstandard_dump_raw (subroutine), 1556
node_component_interoutputstandard_dump_xml (subroutine), 1556
node_component_interoutputstandard_initializor (subroutine), 1556
node_component_interoutputstandard_is_active (function), 1556
node_component_interoutputstandard_name_from_index (subroutine), 1556
node_component_interoutputstandard_ode_step_rates_init (subroutine), 1556
node_component_interoutputstandard_odestepscalesinit (subroutine), 1556
node_component_interoutputstandard_output_count (subroutine), 1556
node_component_interoutputstandard_output_names (subroutine), 1557
node_component_interoutputstandard_read_raw (subroutine), 1557
node_component_interoutputstandard_serialize_rates (subroutine), 1557
node_component_interoutputstandard_serialize_scales (subroutine), 1557
node_component_interoutputstandard_serialize_values (subroutine), 1557
node_component_interoutputstandard_type (function), 1557
node_component_mass_flow_statistics_standard (module), 1708
node_component_mass_flow_statistics_standard_extra_output (subroutine), 1708
node_component_mass_flow_statistics_standard_initialize (subroutine), 1709
node_component_mass_flow_statistics_standard_merger_tree_init (subroutine), 1709
node_component_mass_flow_statistics_standard_rate_compute (subroutine), 1709
node_component_mass_flow_statistics_standard_scale_set (subroutine), 1709
node_component_massflowstatistics_builder (subroutine), 1557
node_component_massflowstatistics_destroy (subroutine), 1557
node_component_massflowstatistics_dump (subroutine), 1557
node_component_massflowstatistics_initializor (subroutine), 1558
node_component_massflowstatistics_move (subroutine), 1558
node_component_massflowstatistics_output (subroutine), 1558
node_component_massflowstatistics_output_count (subroutine), 1558
node_component_massflowstatistics_output_names (subroutine), 1558
node_component_massflowstatistics_remove (subroutine), 1558
node_component_massflowstatistics_type (function), 1558
node_component_massflowstatisticsnull_builder (subroutine), 1558
node_component_massflowstatisticsnull_count (function), 1558
node_component_massflowstatisticsnull_deserialize_rates (subroutine), 1559
node_component_massflowstatisticsnull_deserialize_scales (subroutine), 1559
node_component_massflowstatisticsnull_deserialize_values (subroutine), 1559
node_component_massflowstatisticsnull_destroy (subroutine), 1559
node_component_massflowstatisticsnull_dump (subroutine), 1559
node_component_massflowstatisticsnull_dump_raw (subroutine), 1559
node_component_massflowstatisticsnull_dump_xml (subroutine), 1559
node_component_massflowstatisticsnull_initializor (subroutine), 1559
node_component_massflowstatisticsnull_is_active (function), 1559
node_component_massflowstatisticsnull_name_from_index (subroutine), 1560
node_component_massflowstatisticsnull_ode_step_rates_init (subroutine), 1560
node_component_massflowstatisticsnull_odestepscalesinit (subroutine), 1560
node_component_massflowstatisticsnull_output_count (subroutine), 1560
node_component_massflowstatisticsnull_output_names (subroutine), 1560
node_component_massflowstatisticsnull_read_raw (subroutine), 1560
node_component_massflowstatisticsnull_serialize_rates (subroutine), 1560
node_component_massflowstatisticsnull_serialize_scales (subroutine), 1560
node_component_massflowstatisticsnull_serialize_values (subroutine), 1561
node_component_massflowstatisticsnull_type (function), 1561
node_component_massflowstatisticsstandard_builder (subroutine), 1561

2144
node_component_massflowstatisticsstandard_count (function), 1561
node_component_massflowstatisticsstandard_deserialize_rates (subroutine), 1561
node_component_massflowstatisticsstandard_deserialize_scales (subroutine), 1561
node_component_massflowstatisticsstandard_deserialize_values (subroutine), 1561
node_component_massflowstatisticsstandard_destroy (subroutine), 1561
node_component_massflowstatisticsstandard_dump (subroutine), 1561
node_component_massflowstatisticsstandard_dump_raw (subroutine), 1562
node_component_massflowstatisticsstandard_dump_xml (subroutine), 1562
node_component_massflowstatisticsstandard_initializor (subroutine), 1562
node_component_massflowstatisticsstandard_is_active (function), 1562
node_component_massflowstatisticsstandard_name_from_index (subroutine), 1562
node_component_massflowstatisticsstandard_ode_step_rates_init (subroutine), 1562
node_component_massflowstatisticsstandard_odestepscalesinit (subroutine), 1562
node_component_massflowstatisticsstandard_output_count (subroutine), 1562
node_component_massflowstatisticsstandard_output_names (subroutine), 1563
node_component_massflowstatisticsstandard_read_raw (subroutine), 1563
node_component_massflowstatisticsstandard_serialize_rates (subroutine), 1563
node_component_massflowstatisticsstandard_serialize_scales (subroutine), 1563
node_component_massflowstatisticsstandard_serialize_values (subroutine), 1563
node_component_massflowstatisticsstandard_type (function), 1563
node_component_merging_statistics_recent (module), 1709
node_component_merging_statistics_recent_initialize (subroutine), 1709
node_component_merging_statistics_recent_matches (function), 1710
node_component_merging_statistics_recent_merger_tree_init (subroutine), 1710
node_component_merging_statistics_recent_node_merger (subroutine), 1710
node_component_merging_statistics_recent_node_promotion (subroutine), 1710
node_component_merging_statistics_recent_output (subroutine), 1710
node_component_merging_statistics_recent_output_count (subroutine), 1710
node_component_merging_statistics_recent_output_names (subroutine), 1710
node_component_merging_statistics_standard (module), 1710
node_component_merging_statistics_standard_initialize (subroutine), 1711
node_component_merging_statistics_standard_merger_tree_init (subroutine), 1711
node_component_merging_statistics_standard_merger_tree_init_set (subroutine), 1711
node_component_merging_statistics_standard_node_merger (subroutine), 1711
node_component_merging_statistics_standard_node_promotion (subroutine), 1711
node_component_merging_statistics_standard_satellite_merger (subroutine), 1711
node_component_mergingstatistics_builder (subroutine), 1563
node_component_mergingstatistics_destroy (subroutine), 1563
node_component_mergingstatistics_dump (subroutine), 1563
node_component_mergingstatistics_initializer (subroutine), 1564
node_component_mergingstatistics_move (subroutine), 1564
node_component_mergingstatistics_output (subroutine), 1564
node_component_mergingstatistics_output_count (subroutine), 1564
node_component_mergingstatistics_output_names (subroutine), 1564
node_component_mergingstatistics_remove (subroutine), 1564
node_component_mergingstatistics_type (function), 1564
node_component_mergingstatisticsnull_builder (subroutine), 1564
node_component_mergingstatisticsnull_count (function), 1565
node_component_mergingstatisticsnull_deserialize_rates (subroutine), 1565
node_component_mergingstatisticsnull_deserialize_scales (subroutine), 1565
node_component_mergingstatisticsnull_deserialize_values (subroutine), 1565
node_component_mergingstatisticsnull_destroy (subroutine), 1565
node_component_mergingstatisticsnull_dump (subroutine), 1565
node_component_mergingstatisticsnull_dump_raw (subroutine), 1565
node_component_mergingstatisticsnull_dump_xml (subroutine), 1565
node_component_mergingstatisticsnull_initializor (subroutine), 1565
node_component_mergingstatisticsnull_is_active (function), 1566
node_component_mergingstatisticsnull_name_from_index (subroutine), 1566
node_component_mergingstatisticsnull_ode_step_rates_init (subroutine), 1566
node_component_mergingstatisticsnull_odestepscalesinit (subroutine), 1566
node_component_mergingstatisticsnull_output_count (subroutine), 1566
node_component_mergingstatisticsnull_output_names (subroutine), 1566
node_component_mergingstatisticsnull_read_raw (subroutine), 1566
node_component_mergingstatisticsnull_serialize_rates (subroutine), 1566
node_component_mergingstatisticsnull_serialize_scales (subroutine), 1567
node_component_mergingstatisticsnull_serialize_values (subroutine), 1567
node_component_mergingstatisticsnull_type (function), 1567
node_component_mergingstatisticsrecent_builder (subroutine), 1567
node_component_mergingstatisticsrecent_count (function), 1567
node_component_mergingstatisticsrecent_deserialize_rates (subroutine), 1567
node_component_mergingstatisticsrecent_deserialize_scales (subroutine), 1567
node_component_mergingstatisticsrecent_destroy (subroutine), 1567
node_component_mergingstatisticsrecent_dump (subroutine), 1568
node_component_mergingstatisticsrecent_dump_raw (subroutine), 1568
node_component_mergingstatisticsrecent_dump_xml (subroutine), 1568
node_component_mergingstatisticsrecent_initializor (subroutine), 1568
node_component_mergingstatisticsrecent_is_active (function), 1568
node_component_mergingstatisticsrecent_name_from_index (subroutine), 1568
node_component_mergingstatisticsrecent_ode_step_rates_init (subroutine), 1568
node_component_mergingstatisticsrecent_odestepscalesinit (subroutine), 1568
node_component_mergingstatisticsrecent_output_count (subroutine), 1568
node_component_mergingstatisticsrecent_output_names (subroutine), 1569
node_component_mergingstatisticsrecent_read_raw (subroutine), 1569
node_component_mergingstatisticsrecent_serialize_rates (subroutine), 1569
node_component_mergingstatisticsrecent_serialize_scales (subroutine), 1569
node_component_mergingstatisticsrecent_serialize_values (subroutine), 1569
node_component_mergingstatisticsrecent_type (function), 1569
node_component_mergingstatisticsstandard_builder (subroutine), 1569
node_component_mergingstatisticsstandard_count (function), 1569
node_component_mergingstatisticsstandard_deserialize_rates (subroutine), 1569
node_component_mergingstatisticsstandard_deserialize_scales (subroutine), 1570
node_component_mergingstatisticsstandard_deserialize_values (subroutine), 1570
node_component_mergingstatisticsstandard_destroy (subroutine), 1570
node_component_mergingstatisticsstandard_dump (subroutine), 1570
node_component_mergingstatisticsstandard_dump_raw (subroutine), 1570
node_component_mergingstatisticsstandard_dump_xml (subroutine), 1570
node_component_mergingstatisticsstandard_initializor (subroutine), 1570
node_component_mergingstatisticsstandard_is_active (function), 1570
node_component_mergingstatisticsstandard_name_from_index (subroutine), 1571
node_component_mergingstatisticsstandard_ode_step_rates_init (subroutine), 1571
node_component_mergingstatisticsstandard_odestepscalesinit (subroutine), 1571
node_component_mergingstatisticsstandard_output_count (subroutine), 1571
node_component_mergingstatisticsstandard_output_names (subroutine), 1571
node_component_mergingstatisticsstandard_read_raw (subroutine), 1571
node_component_mergingstatisticsstandard_serialize_rates (subroutine), 1571
node_component_mergingstatisticsstandard_serialize_scales (subroutine), 1571
node_component_mergingstatisticsstandard_serialize_values (subroutine), 1571
node_component_mergingstatisticsstandard_type (function), 1572
node_component_name_from_index (subroutine), 1572
node_component_nbody_builder (subroutine), 1572
node_component_nbody_destroy (subroutine), 1572
node_component_nbody_dump (subroutine), 1572
node_component_nbody_initializor (subroutine), 1572
node_component_nbody_move (subroutine), 1572
node_component_nbody_output (subroutine), 1572
node_component_nbody_output_count (subroutine), 1573
node_component_nbody_output_names (subroutine), 1573
node_component_nbody_remove (subroutine), 1573
node_component_nbody_standard (module), 1711
node_component_nbody_standard_promote (subroutine), 1712
node_component_nbody_type (function), 1573
node_component_nbodynull_builder (subroutine), 1573
node_component_nbodynull_count (function), 1573
node_component_nbodynull_deserialize_rates (subroutine), 1573
node_component_nbodynull_deserialize_scales (subroutine), 1573
node_component_nbodynull_deserialize_values (subroutine), 1573
node_component_nbodynull_destroy (subroutine), 1574
node_component_nbodynull_dump (subroutine), 1574
node_component_nbodynull_dump_raw (subroutine), 1574
node_component_nbodynull_dump_xml (subroutine), 1574
node_component_nbodynull_initializor (subroutine), 1574
node_component_nbodynull_is_active (function), 1574
node_component_nbodynull_name_from_index (subroutine), 1574
node_component_nbodynull_ode_step_rates_init (subroutine), 1574
node_component_nbodynull_odestepscalesinit (subroutine), 1574
node_component_nbodynull_output_count (subroutine), 1575
node_component_nbodynull_output_names (subroutine), 1575
node_component_nbodynull_read_raw (subroutine), 1575
node_component_nbodynull_serialize_rates (subroutine), 1575
node_component_nbodynull_serialize_scales (subroutine), 1575
node_component_nbodynull_serialize_values (subroutine), 1575
node_component_nbodynull_type (function), 1575
node_component_nbodystandard_builder (subroutine), 1575
node_component_nbodystandard_count (function), 1575
node_component_nbodystandard_deserialize_rates (subroutine), 1576
node_component_nbodystandard_deserialize_scales (subroutine), 1576
CODE INDEX

node_component_nbodystandard_deserialize_values (subroutine), 1576
node_component_nbodystandard_destroy (subroutine), 1576
node_component_nbodystandard_dump (subroutine), 1576
node_component_nbodystandard_dump_raw (subroutine), 1576
node_component_nbodystandard_dump_xml (subroutine), 1576
node_component_nbodystandard_initializer (subroutine), 1576
node_component_nbodystandard_is_active (function), 1576
node_component_nbodystandard_name_from_index (subroutine), 1577
node_component_nbodystandard_ode_step_rates_init (subroutine), 1577
node_component_nbodystandard_odestepscalesinit (subroutine), 1577
node_component_nbodystandard_output_count (subroutine), 1577
node_component_nbodystandard_output_names (subroutine), 1577
node_component_nbodystandard_read_raw (subroutine), 1577
node_component_nbodystandard_serialize_rates (subroutine), 1577
node_component_nbodystandard_serialize_scales (subroutine), 1577
node_component_nbodystandard_serialize_values (subroutine), 1577
node_component_nbodystandard_type (function), 1578
node_component_null_double0_inout (function), 1578
node_component_null_void0_inout (subroutine), 1578
node_component_ode_step_initialize_null (subroutine), 1578
node_component_output_count_null (subroutine), 1578
node_component_output_names_null (subroutine), 1578
node_component_output_null (subroutine), 1578
node_component_position_builder (subroutine), 1578
node_component_position_destroy (subroutine), 1578
node_component_position_dump (subroutine), 1579
node_component_position_initializor (subroutine), 1579
node_component_position_move (subroutine), 1579
node_component_position_output (subroutine), 1579
node_component_position_output_count (subroutine), 1579
node_component_position_output_names (subroutine), 1579
node_component_position_preset (module), 1712
node_component_position_preset_node_promotion (subroutine), 1712
node_component_position_remove (subroutine), 1579
node_component_position_type (function), 1579
node_component_positionnull_builder (subroutine), 1579
node_component_positionnull_count (function), 1580
node_component_positionnull_deserialize_rates (subroutine), 1580
node_component_positionnull_deserialize_scales (subroutine), 1580
node_component_positionnull_deserialize_values (subroutine), 1580
node_component_positionnull_destroy (subroutine), 1580
node_component_positionnull_dump (subroutine), 1580
node_component_positionnull_dump_raw (subroutine), 1580
node_component_positionnull_dump_xml (subroutine), 1580
node_component_positionnull_initializor (subroutine), 1580
node_component_positionnull_is_active (function), 1581
node_component_positionnull_name_from_index (subroutine), 1581
node_component_positionnull_ode_step_rates_init (subroutine), 1581
node_component_positionnull_odestepscalesinit (subroutine), 1581
node_component_positionnull_output_count (subroutine), 1581
node_component_positionnull_output_names (subroutine), 1581
node_component_positionnull_read_raw (subroutine), 1581
node_component_positionnull_serialize_rates (subroutine), 1581
node_component_positionnull_serialize_scales (subroutine), 1581
node_component_positionnull_serialize_values (subroutine), 1582
node_component_positionnull_type (function), 1582
node_component_positionpreset_builder (subroutine), 1582
node_component_positionpreset_count (function), 1582
node_component_positionpreset_deserialize_rates (subroutine), 1582
node_component_positionpreset_deserialize_scales (subroutine), 1582
node_component_positionpreset_deserialize_values (subroutine), 1582
node_component_positionpreset_destroy (subroutine), 1582
node_component_positionpreset_dump (subroutine), 1582
node_component_positionpreset_dump_raw (subroutine), 1583
node_component_positionpreset_dump_xml (subroutine), 1583
node_component_positionpreset_initializor (subroutine), 1583
node_component_positionpreset_is_active (function), 1583
node_component_positionpreset_name_from_index (subroutine), 1583
node_component_positionpreset_ode_step_rates_init (subroutine), 1583
node_component_positionpreset_odestepscalesinit (subroutine), 1583
node_component_positionpreset_output_count (subroutine), 1583
node_component_positionpreset_read_raw (subroutine), 1584
node_component_positionpreset_serialize_rates (subroutine), 1584
node_component_positionpreset_serialize_scales (subroutine), 1584
node_component_positionpresetSerialize_values (subroutine), 1584
node_component_positionpreset_type (function), 1584
node_component_potential_null (function), 1584
node_component_rotation_curve_gradient_null (function), 1584
node_component_rotation_curve_null (function), 1584
node_component_satellite_builder (subroutine), 1584
node_component_satellite_destroy (subroutine), 1585
node_component_satellite_dump (subroutine), 1585
node_component_satellite_initializor (subroutine), 1585
node_component_satellite_move (subroutine), 1585
node_component_satellite_orbiting (module), 1712
node_component_satellite_orbiting_create (subroutine), 1712
node_component_satellite_orbiting_initialize (subroutine), 1713
node_component_satellite_orbiting_rate_compute (subroutine), 1713
node_component_satellite_orbiting_scale_set (subroutine), 1713
node_component_satellite_orbiting_time_of_merging (function), 1585
node_component_satellite_orbiting_tree_initialize (subroutine), 1713
node_component_satellite_orbiting_trigger_merger (subroutine), 1713
node_component_satellite_output (subroutine), 1585
node_component_satellite_output_count (subroutine), 1585
node_component_satellite_output_names (subroutine), 1585
node_component_satellite_preset (module), 1713
node_component_satellite_preset_merge_time (function), 1585
node_component_satellite_preset_merge_time_set (subroutine), 1586
node_component_satellite_preset_promote (subroutine), 1713
node_component_satellite_remove (subroutine), 1586
node_component_satellite_standard (module), 1714
node_component_satellite_standard_create (subroutine), 1714
node_component_satellite_standard_halo Formation_task (subroutine), 1714
node_component_satellite_standard_initialize (subroutine), 1714
node_component_satellite_standard_merge_time (function), 1586
node_component_satellite_standard_rate_compute (subroutine), 1714
node_component_satellite_standard_scale_set (subroutine), 1714
node_component_satellite_standard_time_of_merging (function), 1586
node_component_satellite_standard_tree_initialize (subroutine), 1714
node_component_satellite_standard_virial_orbit (function), 1715
node_component_satellite_standard_virial_orbit_set (subroutine), 1715
node_component_satellite_type (function), 1586
node_component_satellite_very_simple (module), 1715
node_component_satellite_very_simple_create (subroutine), 1715
node_component_satellite_very_simple_halo Formation_task (subroutine), 1715
node_component_satellite_very_simple_initialize (subroutine), 1715
node_component_satellite_very_simple_merge_time (function), 1586
node_component_satellite_very_simple_time_of_merging (function), 1586
node_component_satellite_very_simple_tree_initialize (subroutine), 1715
node_component_satellitenull_builder (subroutine), 1586
node_component_satellitenull_count (function), 1586
node_component_satellitenull_deserialize_rates (subroutine), 1587
node_component_satellitenull_deserialize_scales (subroutine), 1587
node_component_satellitenull_deserialize_values (subroutine), 1587
node_component_satellitenull_destroy (subroutine), 1587
node_component_satellitenull_dump (subroutine), 1587
node_component_satellitenull_dump_raw (subroutine), 1587
node_component_satellitenull_dump_xml (subroutine), 1587
node_component_satellitenull_initializor (subroutine), 1587
node_component_satellitenull_is_active (function), 1587
node_component_satellitenull_name_from_index (subroutine), 1588
node_component_satellitenull_ode step_rates_init (subroutine), 1588
node_component_satellitenull_ode stepscalesinit (subroutine), 1588
node_component_satellitenull_output_count (subroutine), 1588
node_component_satellitenull_output_names (subroutine), 1588
node_component_satellitenull_read_raw (subroutine), 1588
node_component_satellitenull_serialize_rates (subroutine), 1588
node_component_satellitenull_serialize_scales (subroutine), 1588
node_component_satellitenull_serialize_values (subroutine), 1588
node_component_satellitenull_type (function), 1589
node_component_satelliteorbiting_builder (subroutine), 1589
node_component_satelliteorbiting_count (function), 1589
node_component_satelliteorbiting_deserialize_rates (subroutine), 1589
node_component_satelliteorbiting_deserialize_scales (subroutine), 1589
node_component_satelliteorbiting_deserialize_values (subroutine), 1589
node_component_satelliteorbiting_destroy (subroutine), 1589
node_component_satelliteorbiting_dump (subroutine), 1589
node_component_satelliteorbiting_dump_raw (subroutine), 1589
node_component_satelliteorbiting_dump_xml (subroutine), 1590
node_component_satelliteorbiting_initializor (subroutine), 1590
node_component_satelliteorbiting_is_active (function), 1590
node_component_satelliteorbiting_name_from_index (subroutine), 1590
node_component_satelliteorbiting_ode_step_rates_init (subroutine), 1590
node_component_satelliteorbiting_odestepscalesinit (subroutine), 1590
node_component_satelliteorbiting_output_count (subroutine), 1590
node_component_satelliteorbiting_output_names (subroutine), 1590
node_component_satelliteorbiting_read_raw (subroutine), 1590
node_component_satelliteorbiting_serialize_rates (subroutine), 1591
node_component_satelliteorbiting_serialize_scales (subroutine), 1591
node_component_satelliteorbiting_serialize_values (subroutine), 1591
node_component_satelliteorbiting_type (function), 1591
node_component_satellitepreset_builder (subroutine), 1591
node_component_satellitepreset_count (function), 1591
node_component_satellitepreset_deserialize_rates (subroutine), 1591
node_component_satellitepreset_deserialize_scales (subroutine), 1591
node_component_satellitepreset_deserialize_values (subroutine), 1591
node_component_satellitepreset_destroy (subroutine), 1592
node_component_satellitepreset_dump (subroutine), 1592
node_component_satellitepreset_dump_raw (subroutine), 1592
node_component_satellitepreset_dump_xml (subroutine), 1592
node_component_satellitepreset_initializor (subroutine), 1592
node_component_satellitepreset_is_active (function), 1592
node_component_satellitepreset_name_from_index (subroutine), 1592
node_component_satellitepreset_ode_step_rates_init (subroutine), 1592
node_component_satellitepreset_odestepscalesinit (subroutine), 1592
node_component_satellitepreset_output_count (subroutine), 1593
node_component_satellitepreset_output_names (subroutine), 1593
node_component_satellitepreset_read_raw (subroutine), 1593
node_component_satellitepreset_serialize_rates (subroutine), 1593
node_component_satellitepreset_serialize_scales (subroutine), 1593
node_component_satellitepreset_serialize_values (subroutine), 1593
node_component_satellitepreset_type (function), 1593
node_component_satellitestandard_builder (subroutine), 1593
node_component_satellitestandard_count (function), 1594
node_component_satellitestandard_deserialize_rates (subroutine), 1594
node_component_satellitestandard_deserialize_scales (subroutine), 1594
node_component_satellitestandard_deserialize_values (subroutine), 1594
node_component_satellitestandard_destroy (subroutine), 1594
node_component_satellitestandard_dump (subroutine), 1594
node_component_satellitestandard_dump_raw (subroutine), 1594
node_component_satellitestandard_dump_xml (subroutine), 1594
node_component_satellitestandard_initializor (subroutine), 1594
node_component_satellitestandard_is_active (function), 1595
node_component_satellitestandard_name_from_index (subroutine), 1595
node_component_satellitestandard_ode_step_rates_init (subroutine), 1595
node_component_satellitestandard_odestepscalesinit (subroutine), 1595
node_component_satellitestandard_output_count (subroutine), 1595
node_component_satellitestandard_output_names (subroutine), 1595
node_component_satellitestandard_read_raw (subroutine), 1595
node_component_satellitestandard_serialize_rates (subroutine), 1595
node_component_satellitestandard_serialize_scales (subroutine), 1595
node_component_satellitestandard_serialize_values (subroutine), 1596
node_component_satellitestandard_type (function), 1596
node_component_satelliteverysimple_builder (subroutine), 1596
node_component_satelliteverysimple_count (function), 1596
node_component_satelliteverysimple_deserialize_rates (subroutine), 1596
node_component_satelliteverysimple_deserialize_scales (subroutine), 1596
node_component_satelliteverysimple_deserialize_values (subroutine), 1596
node_component_satelliteverysimple_destroy (subroutine), 1596
node_component_satelliteverysimple_dump (subroutine), 1596
node_component_satelliteverysimple_dump_raw (subroutine), 1597
node_component_satelliteverysimple_dump_xml (subroutine), 1597
node_component_satelliteverysimple_initializor (subroutine), 1597
node_component_satelliteverysimple_is_active (function), 1597
node_component_satelliteverysimple_name_from_index (subroutine), 1597
node_component_satelliteverysimple_ode_step_rates_init (subroutine), 1597
node_component_satelliteverysimple_odestepscalesinit (subroutine), 1597
node_component_satelliteverysimple_output_count (subroutine), 1597
node_component_satelliteverysimple_output_names (subroutine), 1598
node_component_satelliteverysimple_read_raw (subroutine), 1598
node_component_satelliteverysimple_serialize_rates (subroutine), 1598
node_component_satelliteverysimple_serialize_scales (subroutine), 1598
node_component_satelliteverysimple_serialize_values (subroutine), 1598
node_component_satelliteverysimple_type (function), 1598
node_component_serialize_count_zero (function), 1598
node_component_serialize_null (subroutine), 1598
node_component_spheroid_builder (subroutine), 1598
node_component_spheroid_destroy (subroutine), 1599
node_component_spheroid_dump (subroutine), 1599
node_component_spheroid_initializor (subroutine), 1599
node_component_spheroid_move (subroutine), 1599
node_component_spheroid_output (subroutine), 1599
node_component_spheroid_output_count (subroutine), 1599
node_component_spheroid_output_names (subroutine), 1599
node_component_spheroid_remove (subroutine), 1599
node_component_spheroid_standard (module), 1716
node_component_spheroid_standard_data (module), 1719
node_component_spheroid_standard_density (function), 1599
node_component_spheroid_standard_enclosed_mass (function), 1600
node_component_spheroid_standard_energy_gas_input_rate (subroutine), 1716
node_component_spheroid_standard_half_mass_radius (function), 1600
node_component_spheroid_standard_initialize (subroutine), 1716
node_component_spheroid_standard_initializor (subroutine), 1716
node_component_spheroid_standard_mass_gas_sink_rate (subroutine), 1716
node_component_spheroid_standard_post_evolve (subroutine), 1717
node_component_spheroid_standard_potential (function), 1600
node_component_spheroid_standard_pre_evolve (subroutine), 1717
node_component_spheroid_standard_radius_solve (function), 1717
node_component_spheroid_standard_radius_solve_set (subroutine), 1717
node_component_spheroid_standard_radius_solver (subroutine), 1717
node_component_spheroid_standard_radius_solver_plausibility (subroutine), 1717
node_component_spheroid_standard_rate_compute (subroutine), 1717
node_component_spheroid_standard_rotation_curve (function), 1600
node_component_spheroid_standard_rotation_curve_gradient (function), 1600
node_component_spheroid_standard_satellite_merging (subroutine), 1718
node_component_spheroid_standard_scale_set (subroutine), 1718
node_component_spheroid_standard_star_formation_history_extend (subroutine), 1718
node_component_spheroid_standard_star_formation_history_output (subroutine), 1718
node_component_spheroid_standard_star_formation_history_rate (subroutine), 1718
node_component_spheroid_standard_velocity_solve (function), 1718
node_component_spheroid_standard_velocity_solve_set (subroutine), 1719
node_component_spheroid_type (function), 1600
node_component_spheroidnull_builder (subroutine), 1600
node_component_spheroidnull_count (function), 1601
node_component_spheroidnull_deserialize_rates (subroutine), 1601
node_component_spheroidnull_deserialize_scales (subroutine), 1601
node_component_spheroidnull_deserialize_values (subroutine), 1601
node_component_spheroidnull_destroy (subroutine), 1601
node_component_spheroidnull_dump (subroutine), 1601
node_component_spheroidnull_dump_raw (subroutine), 1601
node_component_spheroidnull_dump_xml (subroutine), 1601
node_component_spheroidnull_initializor (subroutine), 1601
node_component_spheroidnull_is_active (function), 1602
node_component_spheroidnull_name_from_index (subroutine), 1602
node_component_spheroidnull_ode_step_rates_init (subroutine), 1602
node_component_spheroidnull_odesstepcscalesinit (subroutine), 1602
node_component_spheroidnull_output_count (subroutine), 1602
node_component_spheroidnull_output_names (subroutine), 1602
node_component_spheroidnull_read_raw (subroutine), 1602
node_component_spheroidnull_serialize_rates (subroutine), 1602
node_component_spheroidnull_serialize_scales (subroutine), 1602
node_component_spheroidnull_serialize_values (subroutine), 1603
node_component_spheroidnull_type (function), 1603
node_component_spheroidstandard_builder (subroutine), 1603
node_component_spheroidstandard_count (function), 1603
node_component_spheroidstandard_deserialize_rates (subroutine), 1603
node_component_spheroidstandard_deserialize_scales (subroutine), 1603
node_component_spheroidstandard_deserialize_values (subroutine), 1603
node_component_spheroidstandard_destroy (subroutine), 1603
node_component_spheroidstandard_dump (subroutine), 1603
node_component_spheroidstandard_dump_raw (subroutine), 1604
node_component_spheroidstandard_dump_xml (subroutine), 1604
node_component_spheroidstandard_initializor (subroutine), 1604
node_component_spheroidstandard_is_active (function), 1604
node_component_spheroidstandard_name_from_index (subroutine), 1604
node_component_spheroidstandard_ode_step_rates_init (subroutine), 1604
node_component_spheroidstandard_odestepsscalesinit (subroutine), 1604
node_component_spheroidstandard_output_count (subroutine), 1604
node_component_spheroidstandard_output_names (subroutine), 1604
node_component_spheroidstandard_read_raw (subroutine), 1605
node_component_spheroidstandard_serialize_rates (subroutine), 1605
node_component_spheroidstandard_serialize_scales (subroutine), 1605
node_component_spheroidstandard_serialize_values (subroutine), 1605
node_component_spheroidstandard_type (function), 1605
node_component_spin_builder (subroutine), 1605
node_component_spin_destroy (subroutine), 1605
node_component_spin_dump (subroutine), 1605
node_component_spin_initializor (subroutine), 1605
node_component_spin_move (subroutine), 1606
node_component_spin_output (subroutine), 1606
node_component_spin_output_count (subroutine), 1606
node_component_spin_output_names (subroutine), 1606
node_component_spin_preset (module), 1719
node_component_spin_preset3d (module), 1720
node_component_spin_preset3d_initialize (subroutine), 1720
node_component_spin_preset3d_promote (subroutine), 1720
node_component_spin_preset3d_rate_compute (subroutine), 1720
node_component_spin_preset3d_scale_set (subroutine), 1720
node_component_spin_preset_initialize (subroutine), 1719
node_component_spin_preset_promote (subroutine), 1719
node_component_spin_preset_rate_compute (subroutine), 1719
node_component_spin_preset_scale_set (subroutine), 1720
node_component_spin_random (module), 1720
node_component_spin_random_initialize (subroutine), 1721
node_component_spin_random_initialize_spins (subroutine), 1721
node_component_spin_random_promote (subroutine), 1721
node_component_spin_remove (subroutine), 1606
node_component_spin_type (function), 1606
node_component_spinnull_builder (subroutine), 1606
node_component_spinnull_count (function), 1606
node_component_spinnull_deserialize_rates (subroutine), 1607
node_component_spinnull_deserialize_scales (subroutine), 1607
node_component_spinnull_deserialize_values (subroutine), 1607
node_component_spinnull_destroy (subroutine), 1607
node_component_spinnull_dump (subroutine), 1607
node_component_spinnull_dump_raw (subroutine), 1607
node_component_spinnull_dump_xml (subroutine), 1607
node_component_spinnull_initializor (subroutine), 1607
node_component_spinnull_is_active (function), 1607
node_component_spinnull_name_from_index (subroutine), 1608
node_component_spinnull_ode_step_rates_init (subroutine), 1608
node_component_spinnull_odestepscalesinit (subroutine), 1608
node_component_spinnull_output_count (subroutine), 1608
node_component_spinnull_output_names (subroutine), 1608
node_component_spinnull_read_raw (subroutine), 1608
node_component_spinnull_serialize_rates (subroutine), 1608
node_component_spinnull_serialize_scales (subroutine), 1608
node_component_spinnull_serialize_values (subroutine), 1608
node_component_spinnull_type (function), 1609
node_component_spinpreset3d_builder (subroutine), 1609
node_component_spinpreset3d_count (function), 1609
node_component_spinpreset3d_deserialize_rates (subroutine), 1609
node_component_spinpreset3d_deserialize_scales (subroutine), 1609
node_component_spinpreset3d_deserialize_values (subroutine), 1609
node_component_spinpreset3d_destroy (subroutine), 1609
node_component_spinpreset3d_dump (subroutine), 1609
node_component_spinpreset3d_dump_raw (subroutine), 1609
node_component_spinpreset3d_dump_xml (subroutine), 1610
node_component_spinpreset3d_initializor (subroutine), 1610
node_component_spinpreset3d_is_active (function), 1610
node_component_spinpreset3d_name_from_index (subroutine), 1610
node_component_spinpreset3d_ode_step_rates_init (subroutine), 1610
node_component_spinpreset3d_odestepscalesinit (subroutine), 1610
node_component_spinpreset3d_output_count (subroutine), 1610
node_component_spinpreset3d_output_names (subroutine), 1610
node_component_spinpreset3d_read_raw (subroutine), 1610
node_component_spinpreset3d_serialize_rates (subroutine), 1611
node_component_spinpreset3d_serialize_scales (subroutine), 1611
node_component_spinpreset3d_serialize_values (subroutine), 1611
node_component_spinpreset3d_type (function), 1611
node_component_spinpreset_builder (subroutine), 1611
node_component_spinpreset_count (function), 1611
node_component_spinpreset_deserialize_rates (subroutine), 1611
node_component_spinpreset_deserialize_scales (subroutine), 1611
node_component_spinpreset_deserialize_values (subroutine), 1611
node_component_spinpreset_destroy (subroutine), 1612
node_component_spinpreset_dump (subroutine), 1612
node_component_spinpreset_dump_raw (subroutine), 1612
node_component_spinpreset_dump_xml (subroutine), 1612
node_component_spinpreset_initializor (subroutine), 1612
node_component_spinpreset_is_active (function), 1612
node_component_spinpreset_name_from_index (subroutine), 1612
node_component_spinpreset_ode_step_rates_init (subroutine), 1612
node_component_spinpreset_odestepscalesinit (subroutine), 1612
node_component_spinpreset_output_count (subroutine), 1613
node_component_spinpreset_output_names (subroutine), 1613
node_component_spinpreset_read_raw (subroutine), 1613
node_component_spinpreset_serialize_rates (subroutine), 1613
node_component_spinpreset_serialize_scales (subroutine), 1613
node_component_spinpreset_serialize_values (subroutine), 1613
node_component_spinpreset_type (function), 1613
node_component_spinrandom_builder (subroutine), 1613
node_component_spinrandom_count (function), 1613
node_component_spinrandom_deserialize_rates (subroutine), 1614
node_component_spinrandom_deserialize_scales (subroutine), 1614
node_component_spinrandom_deserialize_values (subroutine), 1614
node_component_spinrandom_destroy (subroutine), 1614
node_component_spinrandom_dump (subroutine), 1614
node_component_spinrandom_dump_raw (subroutine), 1614
node_component_spinrandom_dump_xml (subroutine), 1614
node_component_spinrandom_initializor (subroutine), 1614
node_component_spinrandom_is_active (function), 1614
node_component_spinrandom_name_from_index (subroutine), 1615
node_component_spinrandom_ode_step_rates_init (subroutine), 1615
node_component_spinrandom_odestepscalesinit (subroutine), 1615
node_component_spinrandom_output_count (subroutine), 1615
node_component_spinrandom_output_names (subroutine), 1615
node_component_spinrandom_read_raw (subroutine), 1615
node_component_spinrandom_serialize_rates (subroutine), 1615
node_component_spinrandom_serialize_scales (subroutine), 1615
node_component_spinrandom_serialize_values (subroutine), 1615
node_component_spinrandom_type (function), 1616
node_component_surface_density_type (function), 1616
node_components (module), 1678
node_definition_index (function), 1242
node_dump (subroutine), 1616
node_dump_raw (subroutine), 1616
node_dump_xml (subroutine), 1616
node_location (function), 1249
node_lookup (function), 1242
node_output (subroutine), 1616
node_output_count (subroutine), 1616
node_output_names (subroutine), 1616
node_promotion_index_shift (subroutine), 1067
node_promotion_index_shifts (module), 1067
node_property_name_from_index (function), 1616
node_read_raw (subroutine), 1617
node_subhalo_promotion (function), 1067
node_subhalo_promotions (module), 1067
nodecomponent (type), 1617
nodecomponentagestatistics (type), 1617
nodecomponentagestatisticsnull (type), 1617
nodecomponentagestatisticsstandard (type), 1617
nodecomponentbasic (type), 1617
nodecomponentbasicnonevolving (type), 1617
nodecomponentbasicnull (type), 1617
nodecomponentbasicstandard (type), 1617
nodecomponentbasicstandardtracking (type), 1617
nodecomponentblackhole (type), 1618
nodecomponentblackholenull (type), 1618
nodecomponentblackholesimple (type), 1618
nodecomponentblackholestandard (type), 1618
nodecomponentdarkmatterprofile (type), 1618
nodecomponentdarkmatterprofilenull (type), 1618
nodecomponentdarkmatterprofilescaleshape (type), 1618
nodecomponentdarkmatterprofilescalescalepreset (type), 1618
nodecomponentdarkmatterprofilescalescale (type), 1618
nodecomponentdisk (type), 1618
nodecomponentdisknull (type), 1619
nodecomponentdiskverysimple (type), 1619
nodecomponentdynamicsstatistics (type), 1619
nodecomponentdynamicsstatisticsbars (type), 1619
nodecomponentdynamicsstatisticsnull (type), 1619
nodecomponentformationtime (type), 1619
nodecomponentformationtimecole2000 (type), 1619
nodecomponentformationtimenull (type), 1619
nodecomponenthosthistory (type), 1620
nodecomponenthosthistorynull (type), 1620
nodecomponenthosthistorystandard (type), 1620
nodecomponenthosthalo (type), 1620
nodecomponenthosthalocoldmode (type), 1620
nodecomponenthosthalonull (type), 1620
nodecomponenthosthalooutflowtracking (type), 1620
nodecomponenthosthalostandard (type), 1620
nodecomponenthosthaloverysimple (type), 1620
nodecomponentindices (type), 1620
nodecomponentindicesnull (type), 1621
nodecomponentindicesstandard (type), 1621
nodecomponentinteroutput (type), 1621
nodecomponentinteroutputnull (type), 1621
nodecomponentinteroutputstandard (type), 1621
nodecomponentmassflowstatistics (type), 1621
nodecomponentmassflowstatisticsnull (type), 1621
nodecomponentmassflowstatisticsstandard (type), 1621
nodecomponentmergingstatistics (type), 1621
nodecomponentmergingstatisticsnull (type), 1621
nodecomponentmergingstatisticsrecent (type), 1622
nodecomponentmergingstatisticsstandard (type), 1622
nodecomponentnbody (type), 1622
nodecomponentnbodynull (type), 1622
nodecomponentnbodystandard (type), 1622
nodecomponentposition (type), 1622
nodecomponentpositionnull (type), 1622
nodecomponentpositionpreset (type), 1622
nodecomponentssatellite (type), 1622
nodecomponentsatellitenull (type), 1622
nodecomponentsatelliteorbing (type), 1623
nodecomponentsatellitepreset (type), 1623
nodecomponentsatellitestandard (type), 1623
nodecomponentsatelliteverysimple (type), 1623
nodecomponentspheroid (type), 1623
nodecomponentspheroidnull (type), 1623
nodecomponentspheroidstandard (type), 1623
nodecomponentspin (type), 1623
nodecomponentspinnull (type), 1623
nodecomponentspinpreset (type), 1624
nodecomponentspinpreset3d (type), 1624
nodecomponentspinrandom (type), 1624
nodedata (type), 1259
nodedataabundanceccscalarevolvable (type), 1624
nodedatachemicalabundanceccscalarevolvable (type), 1624
nodedatadoublen1d (type), 1624
nodedatadoublen1devolvable (type), 1624
nodedatadoublescalar (type), 1624
nodedatadoublescalarevolvable (type), 1624
nodedatagalacticus (type), 1259, 1271
nodedatahistoryscalar (type), 1624
nodedatahistoryscalarevolvable (type), 1625
nodedataid (type), 1625
nodedataintegerscalar (type), 1625
nodedatakeplerorbitscalar (type), 1625
nodedatalogicalscalar (type), 1625
nodedatalongintegerhistoryscalar (type), 1625
nodedatalongintegerscalar (type), 1625
nodedatastellarluminositiesscalarevolvable (type), 1625
nodedatassussing (type), 1259, 1264
nodedatatensorrank2dimension3symmetricscalarevolvable (type), 1625
nodeevent (type), 1626
nullaccretedmass (function), 893, 899
nullaccretedmasschemicals (function), 893, 899
nullaccretedmassmetals (function), 893, 899
nullaccretionrate (function), 893, 899
nullaccretionratechemicals (function), 893, 899
nullaccretionratemetals (function), 893, 899
nullconstructor (function), 893, 900
nulldefaultconstructor (function), 1166, 1174, 1762, 1771
nulldensity (function), 1166, 1174
nulldensitylogslope (function), 1167, 1174
nulldestructor (subroutine), 1167, 1174, 1763, 1771
nullenclosedmass (function), 1167, 1174
nullfailedaccretedmass (function), 893, 900
nullfailedaccretionrate (function), 894, 900
nullradialmoment (function), 1167, 1175
nullrotationnormalization (function), 1167, 1175
nulltimeuntilmerging (function), 1763, 1771
number_density_critical_over_number_density_hydrogen (subroutine), 955
numerical.comparison.F90 (file), 1293
numerical.constants.astronomical.F90 (file), 1295
numerical.constants.atomic.F90 (file), 1298
numerical.constants.boolean.F90 (file), 1298
numerical.constants.math.F90 (file), 1299
numerical.constants.physical.F90 (file), 1302
numerical.constants.prefixes.F90 (file), 1305
numerical.constants.units.F90 (file), 1306
numerical.fftw3.F90 (file), 1307
numerical.integration.F90 (file), 1307
numerical.interpolation.2D.irregular.F90 (file), 1309
numerical.interpolation.F90 (file), 1309
numerical.meshes.F90 (file), 1313
numerical.ODE_solver.error_codes.F90 (file), 1293
numerical.ODE_solver.F90 (file), 1288
numerical.ODE_solver.ODEIV2.F90 (file), 1289
numerical.ODE_solver.ODEIV2.wrapper.F90 (file), 1289
numerical.random.F90 (file), 1313
numerical.random.gaussian.F90 (file), 1315
numerical.random.quasi.F90 (file), 1315
numerical.ranges.F90 (file), 1316
numerical.root_finder.F90 (file), 1317
numerical.search.F90 (file), 1319
numerical.sort.F90 (file), 1320
numerical.comparison (module), 1293
numerical.constants.astronomical (module), 1295
numerical.constants.atomic (module), 1298
numerical.constants.boolean (module), 1299
numerical.constants.math (module), 1299
numerical.constants.physical (module), 1302
numerical.constants.prefixes (module), 1305
numerical.constants.units (module), 1306
numerical.integration (module), 1307
numerical.interpolation (module), 1309
numerical.interpolation_2d_irregular (module), 1309
numerical.ranges (module), 1316
objects.abundances.F90 (file), 1322
objects.chemical_abundances.F90 (file), 1329
objects.chemical_structure.F90 (file), 1333
objects.coordinates.F90 (file), 1334
objects.history.F90 (file), 1338
objects.kepler_orbits.F90 (file), 1344
objects.mass_distributions.F90 (file), 1349
objects.merger_tree_data.F90 (file), 1356
objects.merger_trees.dump.F90 (file), 1360
objects.merger_trees.information_content.F90 (file), 1362
objects.nodes.bindings.C.F90 (file), 1671
objects.nodes.components.age_statistics.standard.F90 (file), 1678
ode_solver_free (subroutine), 1289
odeiv2_solve (subroutine), 1289
odeiv2_solver (module), 1289
odeiv2_solver_free (subroutine), 1289
omegabaryonsimple (function), 1001, 1003
omegacurvaturesimple (function), 1001, 1003
omegadarkenergysimple (function), 1001, 1003
omegamattersimple (function), 1002, 1003
omegaradiationsimple (function), 1002, 1003
op_assign_ch_vs (subroutine), 1218
op_assign_vs_ch (subroutine), 1218
op_concat_ch_vs (function), 1218
op_concat_vs_ch (function), 1219
op_concat_vs_vs (function), 1219
op_eq_ch_vs (function), 1219
op_eq_vs_ch (function), 1219
op_eq_vs_vs (function), 1219
operator (interface), 1220
operator(*) (interface), 1329, 1344, 1725, 1736
operator(.intersection.) (interface), 1933
operator(//) (interface), 1220, 1981
operator(/=) (interface), 1220
operator(==) (interface), 1220
operator(*=) (interface), 1220
optimal_sampling.stellar_mass_function.F90 (file), 1739
optimal_sampling_smf (program), 1739
orbit_construct (function), 1249
output_times_initialize (subroutine), 1151
outputgroup (type), 1124
perform_node_events (subroutine), 1275
perform_tree_events (subroutine), 1276
perturbation_dynamics_solver (subroutine), 1877
perturbation_integrand (function), 1879
perturbation_maximum_radius (function), 1879
perturbationnodes (function), 1877
CODE INDEX

phase_space_position_realize (subroutine), 1249
photoionization_heating_rate_integrand (function), 1196
photoionization_rate_integrand (function), 1196
piecewise_power_law_imf_normalize (subroutine), 1800
piecewise_power_law_imf_phi (interface), 1800
piecewise_power_law_imf_phi_array (function), 1800
piecewise_power_law_imf_phi_scalar (function), 1800
pointed_at_node (function), 1273
positioncountlinked (function), 1626
positioncreatebyinterrupt (subroutine), 1626
positioncreatelinked (subroutine), 1626
positiondestroylinked (subroutine), 1626
positionget (function), 1626
positionnullbindingdouble1inout (function), 1626
positionnullbindinghistory0inout (function), 1626
positionnullbindinginteger0in (function), 1626
positionnullbindingratedouble1inout (subroutine), 1627
positionnullbindingratehistory0inout (subroutine), 1627
positionnullbindingrateinteger0in (subroutine), 1627
positionnullbindingsetdouble1inout (subroutine), 1627
positionnullbindingsethistory0inout (subroutine), 1627
positionnullbindingsetinteger0in (subroutine), 1627
positionposition (function), 1627
positionpositionattributematch (function), 1627
positionpositionhistory (function), 1627
positionpositionhistoryattributematch (function), 1628
positionpositionhistoryisgettable (function), 1628
positionpositionisgettable (function), 1628
positionresetposition (function), 1628
positionresetpositionhistoryget (function), 1628
positionresetpositionhistoryset (subroutine), 1628
positionresetpositionset (subroutine), 1628
positionvelocity (function), 1629
positionvelocityattributematch (function), 1629
positionvelocityisgettable (function), 1629
postprocessor (type), 1837
postprocessors (type), 1837
power_spectra (module), 1868
power_spectra (program), 883
Power_Spectra.F90 (file), 883
power_spectra.tasks.F90 (file), 1739
power_spectra_nonlinear (module), 1870
power_spectra_nonlinear_cosmicemu (module), 1870
power_spectra_nonlinear_linear (module), 1871
power_spectra_nonlinear_peacockdodds1996 (module), 1871
power_spectrum (function), 1869
power_spectrum_close_file (subroutine), 1739
power_spectrum_compute (subroutine), 1740
power_spectrum_dimensionless (function), 1869
power_spectrum_initialize (subroutine), 1872
power_spectrum_logarithmic_derivative (function), 1869
power_spectrum_nonlinear (function), 1870
power_spectrum_nonlinear_cosmicemu (function), 1870
power_spectrum_nonlinear_cosmicemu_initialize (subroutine), 1870
power_spectrum_nonlinear_initialize (subroutine), 1871
power_spectrum_nonlinear_linear (function), 1872
power_spectrum_nonlinear_linear_initialize (subroutine), 1872
power_spectrum_nonlinear_peacockdodds1996 (function), 1871
power_spectrum_nonlinear_peacockdodds1996_initialize (subroutine), 1871
power_spectrum_open_file (subroutine), 1740
power_spectrum_output (subroutine), 1740
power_spectrum_power_law_tabulate (subroutine), 1873
power_spectrum_tasks (module), 1739
power_spectrum_window_function (function), 1874
power_spectrum_window_function_sharp_kspace (function), 1875
power_spectrum_window_function_th_kss_hybrid (function), 1876
power_spectrum_window_function_top_hat (function), 1875
power_spectrum_window_function_wavenumber_maximum (function), 1874
power_spectrum_window_function_wavenumber_maximum_sharp_kspace (function), 1875
power_spectrum_window_function_wavenumber_maximum_th_kss_hybrid (function), 1876
power_spectrum_window_function_wavenumber_maximum_top_hat (function), 1875
power_spectrum_window_functions (module), 1874
power_spectrum_window_functions_initialize (subroutine), 1874
power_spectrum_window_functions_sharp_kspace (module), 1874
power_spectrum_window_functions_sharp_kspace_initialize (subroutine), 1875
power_spectrum_window_functions_th_kss_hybrid (module), 1876
power_spectrum_window_functions_th_kss_hybrid_initialize (subroutine), 1876
power_spectrum_window_functions_top_hat (module), 1875
power_spectrum_window_functions_top_hat_initialize (subroutine), 1876
prada2011b0 (function), 1055, 1060
prada2011b1 (function), 1055, 1060
prada2011c (function), 1056, 1060
prada2011cmin (function), 1056, 1060
prada2011concentration (function), 1056, 1060
prada2011constructor (function), 1056, 1061
prada2011darkmatterprofiledefinition (function), 1056, 1061
prada2011defaultconstructor (function), 1056, 1061
prada2011densitycontrastdefinition (function), 1056, 1061
prada2011 inversesigmin (function), 1056, 1061
presetdefaultconstructor (function), 1763, 1772
presetdefaultstructor (subroutine), 1763, 1772
presentsimeuntilmerging (function), 1763, 1772
primordial_power_spectra (module), 1872
primordial_power_spectra_transferred (module), 1873
primordial_power_spectrum (function), 1872
primordial_power_spectrum_logarithmic_derivative (function), 1872

2163
CODE INDEX

radiation_igb_file_set (subroutine), 1742
radiation_igb_internal_flux (subroutine), 1743
radiation_igb_internal_initialize (subroutine), 1743
radiation_igb_internal_set (subroutine), 1743
radiation_initialize_intergalactic_background (subroutine), 1741
radiation_integrate_over_cross_section (function), 1723
radiation_intergalactic_background (module), 1741
radiation_intergalactic_background_internal (module), 1742
radiation_intergalactic_background_internal_initialize (subroutine), 1743
radiation_intergalactic_background_internal_update (function), 1743
radiation_is_defined (function), 1723
radiation_null (module), 1744
radiation_set cmb (subroutine), 1740
radiation_set_intergalactic_background (subroutine), 1741
radiation_set_node (subroutine), 1723
radiation_set_null (subroutine), 1744
radiation_set_time (subroutine), 1724
radiation_set_time cmb (subroutine), 1740
radiation_set_time_intergalactic_background (subroutine), 1741
radiation_set_time_null (subroutine), 1744
radiation_structure (module), 1721
radiation_temperature (function), 1724
radiation_temperature cmb (subroutine), 1741
radiation_temperature_intergalactic_background (subroutine), 1741
radiation_temperature_null (subroutine), 1744
radiation_time (function), 1724
radiationdata (type), 1724
radiationstructure (type), 1724
radiusperturbation (function), 1877
radiusspecifier (type), 1138, 1148
ram_pressure_stripping.mass_loss_rate.disks.F90 (file), 1745
ram_pressure_stripping.mass_loss_rate.disks.null.F90 (file), 1745
ram_pressure_stripping.mass_loss_rate.disks.simple.F90 (file), 1745
ram_pressure_stripping.mass_loss_rate.spheroids.F90 (file), 1746
ram_pressure_stripping.mass_loss_rate.spheroids.null.F90 (file), 1747
ram_pressure_stripping.mass_loss_rate.spheroids.simple.F90 (file), 1747
ram_pressure_stripping_mass_loss_rate_disk (function), 1745
ram_pressure_stripping_mass_loss_rate_disk_null (function), 1745
ram_pressure_stripping_mass_loss_rate_disk_simple (function), 1746
ram_pressure_stripping_mass_loss_rate_disks (module), 1745
ram_pressure_stripping_mass_loss_rate_disks_null (module), 1745
ram_pressure_stripping_mass_loss_rate_disks_null_init (subroutine), 1745
ram_pressure_stripping_mass_loss_rate_disks_simple (module), 1746
ram_pressure_stripping_mass_loss_rate_disks_simple_init (subroutine), 1746
ram_pressure_stripping_mass_loss_rate_spheroid (function), 1746
ram_pressure_stripping_mass_loss_rate_spheroid_null (function), 1747
ram_pressure_stripping_mass_loss_rate_spheroid_simple (function), 1747
ram_pressure_stripping_mass_loss_rate_spheroids (module), 1746
ram_pressure_stripping_mass_loss_rate_spheroids_null (module), 1747
CODE INDEX

ram_pressure_stripping_mass_loss_rate_spheroids_null_init (subroutine), 1747
ram_pressure_stripping_mass_loss_rate_spheroids_simple (module), 1747
ram_pressure_stripping_mass_loss_rate_spheroids_simple_init (subroutine), 1749
recentapply (subroutine), 1837, 1841
recentdefaultconstructor (function), 1837, 1841
recentgenericconstructor (function), 1837, 1841
recfastconstructor (function), 1190, 1191
recfastdefaultconstructor (function), 1190, 1191
recycled_fraction_integrand (function), 1790
regex (interface), 1975
regular_expression_constructor (function), 1975
regular_expression_destroy (subroutine), 1975
regular_expression_destructor (subroutine), 1975
regular_expression_match (function), 1975
regular_expressions (module), 1975
remove (interface), 1221
remove_ch (function), 1221
remove_vs (function), 1222
repeat (interface), 1222
repeat_ (function), 1222
replace (interface), 1222
replace_ch_ch_auto (function), 1222
replace_ch_ch_ch_target (function), 1222
replace_ch_ch_fixed (function), 1222
replace_ch_ch_vs_target (function), 1222
replace_ch_vs_auto (function), 1222
replace_ch_vs_ch_target (function), 1222
replace_ch_vs_fixed (function), 1222
replace_ch_vs_vs_target (function), 1222
replace_vs_ch_auto (function), 1223
replace_vs_ch_ch_target (function), 1223
replace_vs_ch_fixed (function), 1223
replace_vs_ch_vs_target (function), 1223
replace_vs_vs_auto (function), 1223
replace_vs_vs_ch_target (function), 1223
replace_vs_vs_fixed (function), 1223
replace_vs_vs_vs_target (function), 1223
reset_records (subroutine), 1281
ricotti2000defaultconstructor (function), 1167, 1168
ricotti2000initialize (subroutine), 1167, 1168
rlss (function), 1196
root_finder (module), 1317
root_finder_find (function), 1318
root_finder_is_initialized (function), 1318
root_finder_range_expand (subroutine), 1318
root_finder_root_function (subroutine), 1318
root_finder_tolerance (subroutine), 1318
root_finder_type (subroutine), 1319
root_finder_wrapper_function (function), 1319
root_function_1 (function), 1907
root_function_2 (function), 1907
root_function_3 (function), 1907
rootfinder (type), 1319
rowstructure (type), 1939

satellite_dynamical_friction (module), 1749
satellite_dynamical_friction_acceleration (function), 1750
satellite_dynamical_friction_acceleration_chandrasekhar (function), 1749
satellite_dynamical_friction_acceleration_initialize (subroutine), 1750
satellite_dynamical_friction_acceleration_null (function), 1750
satellite_dynamical_friction_acceleration_null_initialize (subroutine), 1750
satellite_dynamical_friction_chandrasekhar_initialize (subroutine), 1749
satellite_merger_process (subroutine), 1281
satellite_merging_mass_movement (subroutine), 1751
satellite_merging_mass_movement_baugh2005 (subroutine), 1751
satellite_merging_mass_movement_simple (subroutine), 1752
satellite_merging_mass_movement_store (subroutine), 1751
satellite_merging_mass_movement_variable (subroutine), 1753
satellite_merging_mass_movements (module), 1751
satellite_merging_mass_movements_baugh2005 (module), 1750
satellite_merging_mass_movements_baugh2005_initialize (subroutine), 1751
satellite_merging_mass_movements_descriptors (module), 1752
satellite_merging_mass_movements_simple (module), 1752
satellite_merging_mass_movements_simple_initialize (subroutine), 1752
satellite_merging_mass_movements_variable (module), 1753
satellite_merging_mass_movements_variable_initialize (subroutine), 1753
satellite_merging_remnant_progenitor_properties (subroutine), 1756
satellite_merging_remnant_progenitor_properties_cole2000 (subroutine), 1756
satellite_merging_remnant_progenitor_properties_cole2000_init (subroutine), 1756
satellite_merging_remnant_progenitor_properties_standard (subroutine), 1757
satellite_merging_remnant_progenitor_properties_standard_init (subroutine), 1757
satellite_merging_remnant_progenitors_properties_cole2000 (module), 1755
satellite_merging_remnant_progenitors_properties_cole2000_init (module), 1757
satellite_merging_remnant_size (subroutine), 1755
satellite_merging_remnant_size_cole2000 (subroutine), 1753
satellite_merging_remnant_size_coving2008 (subroutine), 1754
satellite_merging_remnant_size_null (subroutine), 1755
satellite_merging_remnant_sizes (module), 1754
satellite_merging_remnant_sizes_cole2000 (module), 1753
satellite_merging_remnant_sizes_cole2000_initialize (subroutine), 1754
satellite_merging_remnant_sizes_coving2008 (module), 1754
satellite_merging_remnant_sizes_coving2008_initialize (subroutine), 1754
satellite_merging_remnant_sizes_null (module), 1755
satellite_merging_remnant_sizes_null_initialize (subroutine), 1755
satellite_merging_remnant_sizes_progenitors (module), 1756
satellite_merging_remnant_sizes_properties (module), 1757
satellite_merging_timescale_state_retrieve (subroutine), 1763
satellite_merging_timescale_state_snapshot (subroutine), 1763
satellite_merging_timescale_state_store (subroutine), 1763
satellite_merging_timescales (module), 1758
satellite_move_to_new_host (subroutine), 1778
satellite_orbit_convert_to_current_potential (function), 1776
satellite_orbit_equivalent_circular_orbit_radius (function), 1778
satellite_orbit_extremum_phase_space_coordinates (subroutine), 1778
satellite_orbits (module), 1776
satellite_promotion (module), 1778
satellite_tidal_field (function), 1779
satellite_tidal_heating (module), 1780
satellite_tidal_heating_rate (function), 1780
satellite_tidal_heating_rate_gnedin (function), 1781
satellite_tidal_heating_rate_gnedin_initialize (subroutine), 1781
satellite_tidal_heating_rate_initialize (subroutine), 1780
satellite_tidal_heating_rate_null (function), 1781
satellite_tidal_heating_rate_null_initialize (subroutine), 1781
satellite_tidal_heating_rate_null_initialize (subroutine), 1780
satellite_tidal_stripping (module), 1781
satellite_tidal_stripping_initialize (subroutine), 1782
satellite_tidal_stripping_rate (function), 1782
satellite_tidal_stripping_rate_null (function), 1783
satellite_tidal_stripping_rate_null_initialize (subroutine), 1783
satellite_tidal_stripping_rate_zentner2005 (function), 1782
satellite_tidal_stripping_rate_zentner2005_initialize (subroutine), 1782
satellite_boundmass (function), 1629
satellite_boundmass_attribute_match (function), 1629
satellite_boundmass_history (function), 1629
satellite_boundmass_history_attribute_match (function), 1629
satellite_boundmass_history_is_gettable (function), 1629
satellite_boundmass_is_gettable (function), 1630
satellite_count_linked (function), 1630
satellite_create_by_interrupt (subroutine), 1630
satellite_create_linked (subroutine), 1630
satellite_destroy_linked (subroutine), 1630
satellite_get (function), 1630
satellite_merge_time (function), 1630
satellite_merge_time_attribute_match (function), 1630
satellite_merge_time_is_gettable (function), 1631
satellite_merging_timescales (interface), 1763
satellite_merging_timescales_boylan_kolchin2008 (interface), 1758, 1764
satellite_merging_timescales_class (type), 1764
satellite_merging_timescales_constructor_default (function), 1764
satellite_merging_timescales_constructor_named (function), 1764
satellite_merging_timescales_infinite (interface), 1764, 1771
satellite_merging_timescales_initialize (subroutine), 1764
satellite_merging_timescales_is_finalizable (function), 1764
satellite_merging_timescales_jiang2008 (interface), 1764, 1767
satellite_merging_timescales_lacey_cole1993 (interface), 1764, 1768
satellite_merging_timescales_lacey_cole1993_tormen (interface), 1765, 1769
satellite_merging_timescales_null (interface), 1765, 1772
satellite_merging_timescales_preset (interface), 1765, 1772
satellitemergingtimescalesstatussatterstore (subroutine), 1765
satellitemergingtimescalesstatussnapshot (subroutine), 1765
satellitemergingtimescalesstatusstore (subroutine), 1765
satellitemergingtimescalesstatusuntilmergingnull (function), 1765
satellitemergingtimescalesstatusvillalobos2013 (interface), 1765, 1769
satellitemergingtimescaleswetzelwhite2010 (interface), 1765, 1770
satellitenodeindex (function), 1631
satellitenodeindexattributematch (function), 1631
satellitenodeindexhistory (function), 1631
satellitenodeindexhistoryattributematch (function), 1631
satellitenodeindexhistoryisgettable (function), 1631
satellitenodeindexisgettable (function), 1631
satellitenullbindingdouble0inout (function), 1631
satellitenullbindingdouble1inout (function), 1631
satellitenullbindinghistory0inout (function), 1632
satellitenullbindinginteger0in (function), 1632
satellitenullbindingkeplerorbit0inout (function), 1632
satellitenullbindinglonginteger0inout (function), 1632
satellitenullbindinglongintegerhistory0inout (function), 1632
satellitenullbindingratedouble0inout (subroutine), 1632
satellitenullbindingratedouble1inout (subroutine), 1632
satellitenullbindingratehistory0inout (subroutine), 1632
satellitenullbindingrateinteger0in (subroutine), 1632
satellitenullbindingratekeplerorbit0inout (subroutine), 1632
satellitenullbindingratelonginteger0inout (subroutine), 1633
satellitenullbindingratelongintegerhistory0inout (subroutine), 1633
satellitenullbindingratetensorrank2dimension3symmetric0inout (subroutine), 1633
satellitenullbindingsetdouble0inout (subroutine), 1633
satellitenullbindingsetdouble1inout (subroutine), 1633
satellitenullbindingsethistory0inout (subroutine), 1633
satellitenullbindingsetinteger0in (subroutine), 1633
satellitenullbindingsetteplerorbit0inout (subroutine), 1633
satellitenullbindingsetlonginteger0inout (subroutine), 1633
satellitenullbindingsetlongintegerhistory0inout (subroutine), 1633
satellitenullbindingsettensorrank2dimension3symmetric0inout (subroutine), 1634
satellitenullbindingsettensorrank2dimension3symmetric0inout (function), 1634
satelliteorbitingboundmasscount (function), 1634
satelliteorbitingboundmassget (function), 1634
satelliteorbitingboundmassrate (subroutine), 1634
satelliteorbitingboundmassscale (subroutine), 1634
satelliteorbitingboundmassset (subroutine), 1634
satelliteorbitingmergetimeget (function), 1634
satelliteorbitingmergetimeset (subroutine), 1634
satelliteorbitingpositioncount (function), 1635
satelliteorbitingpositionget (function), 1635
satelliteorbitingpositionrate (subroutine), 1635
satelliteorbitingpositionscale (subroutine), 1635
satelliteorbitingpositionset (subroutine), 1635
satelliteorbitingtidalheatingnormalizedcount (function), 1635
satelliteorbitingtidalheatingnormalizedget (function), 1635
satelliteorbitingtidalheatingnormalizedrate (subroutine), 1635
satelliteorbitingtidalheatingnormalizedscale (subroutine), 1636
satelliteorbitingtidalheatingnormalizedset (subroutine), 1636
satelliteorbitingtidaltensorpathintegratedcount (function), 1636
satelliteorbitingtidaltensorpathintegratedget (function), 1636
satelliteorbitingtidaltensorpathintegratedrate (subroutine), 1636
satelliteorbitingtidaltensorpathintegratedscale (subroutine), 1636
satelliteorbitingtidaltensorpathintegratedset (subroutine), 1636
satelliteorbitingvelocitycount (function), 1636
satelliteorbitingvelcityget (function), 1637
satelliteorbitingvelocityrate (subroutine), 1637
satelliteorbitingvelcityscale (subroutine), 1637
satelliteorbitingvelocityset (subroutine), 1637
satelliteorbitingvirialorbitget (function), 1637
satelliteorbitingvirialorbitset (function), 1637
satelliteposition (function), 1637
satellitepositionattributematch (function), 1637
satellitepositionisgettable (function), 1637
satellitepresetboundmasshistoryget (function), 1638
satellitepresetboundmasshistoryset (subroutine), 1638
satellitepresetmergeboundmass (function), 1638
satellitepresetnodeindex (function), 1638
satellitepresetnodeindexhistoryget (function), 1638
satellitepresetnodeindexhistoryset (subroutine), 1638
satellitepresettimeofmergingget (function), 1638
satellitepresettimeofmergingset (subroutine), 1638
satellitepresetvirialorbitget (function), 1639
satellitepresetvirialorbitset (subroutine), 1639
satellites.dynamical_friction.acceleration.Chandrasekhar1943.F90 (file), 1749
satellites.dynamical_friction.acceleration.F90 (file), 1749
satellites.dynamical_friction.acceleration.null.F90 (file), 1750
satellites.merging.mass_movements.descriptors.F90 (file), 1752
satellites.merging.mass_movements.F90 (file), 1751
satellites.merging.mass_movements.simple.F90 (file), 1752
satellites.merging.mass_movements.very_simple.F90 (file), 1753
satellites.merging.remnant_sizes.Cole2000.F90 (file), 1753
satellites.merging.remnant_sizes.Covington2008.F90 (file), 1754
satellites.merging.remnant_sizes.F90 (file), 1754
satellites.merging.remnant_sizes.null.F90 (file), 1755
satellites.merging.remnant_sizes.progenitor_properties.Cole2000.F90 (file), 1755
satellites.merging.remnant_sizes.progenitor_properties.F90 (file), 1756
satellites.merging.remnant_sizes.progenitor_properties.standard.F90 (file), 1757
satellites.merging.remnant_sizes.properties.F90 (file), 1757
satellites.merging.timescale.F90 (file), 1758
satellites.merging.timescale.infinite.F90 (file), 1771
satellitetidal_tensor_path_integrated (function), 1641
satellitetidal_tensor_path_integrated_attributematch (function), 1641
satellitetidal_tensor_path_integrated_is_gettable (function), 1641
satellite_time_of_merging (function), 1641
satellite_time_of_merging_attributematch (function), 1642
satellite_time_of_merging_is_gettable (function), 1642
satellite_velocity (function), 1642
satellite_velocity_attributematch (function), 1642
satellite_velocity_is_gettable (function), 1642
satellite_velocity_time_of_merging (function), 1642
satellite_velocity_time_of_merging_attributematch (function), 1642
satellite_velocity_time_of_merging_is_gettable (function), 1642
satellite_velocity_time_of_merging_gettimecount (function), 1642
satellite_velocity_time_of_merging_gettimerate (subroutine), 1642
satellite_velocity_time_of_merging_gettimescale (subroutine), 1642
satellite_velocity_time_of_merging_gettimeset (subroutine), 1643
satellite_velocity_time_of_merging_gettimescale (subroutine), 1643
satellite_time_of_merging_attributematch (function), 1643
satellite_time_of_merging_is_gettable (function), 1643
scan (interface), 1223
scan_ch_vs (function), 1223
scan_for_branch_jumps (subroutine), 1250
scan_for_mergers (subroutine), 1250
scan_for_subhalo_promotions (subroutine), 1250
scan_vs_ch (function), 1223
scan_vs_vs (function), 1223
search_array (interface), 1319
search_array_double (function), 1319
search_array_for_closest (function), 1320
search_array_integer8 (function), 1320
search_array_varstring (function), 1320
search_indexed (interface), 1320
search_indexed_integer8 (function), 1320
semaphore (type), 1976
semaphore_close (subroutine), 1976
semaphore_open (function), 1976
semaphore_post (subroutine), 1976
semaphore_post_on_error (subroutine), 1976
semaphore_unlink (subroutine), 1976
semaphore_wait (subroutine), 1976
semaphorelist (type), 1976
semaphores (module), 1975
serializetoarraycount (function), 1643
serializetoarrayrates (subroutine), 1643
serializetoarrayscales (subroutine), 1643
serializetoarrayvalues (subroutine), 1643
sersic_abel_integrand (function), 1355
sersic_coefficient_root (function), 1355
sersic_profile_tabulate (subroutine), 1355
sersic_profile_tabulate_state_retrieve (subroutine), 1355
sersic_profile_tabulate_state_store (subroutine), 1356
set_coefficients (subroutine), 920
set_integer_scalar_ch (subroutine), 1937
set_integer_scalar_vs (subroutine), 1937
set_memory_prefix (subroutine), 1974
shocks.one_dimensional.F90 (file), 1783
shocks_1d (module), 1783
shocks_1d_density_jump (function), 1783
sigma_8 (function), 1869
simple_merger_tree_file_maker (program), 884
Simple_Merger_Tree_File_Maker.F90 (file), 884
simpleaccretedmass (function), 894, 900
simpleaccretedmasschemicals (function), 894, 900
simpleaccretedmassmetals (function), 894, 900
simpleaccretionrate (function), 894, 901
simpleaccretionratechemicals (function), 894, 901
simpleaccretionratemetals (function), 894, 901
simpleaccretionratetotal (function), 894, 901
simplechemicalmasses (function), 895, 901
simpleconstructor (function), 895, 901, 1190, 1197
simpledefaultconstructor (function), 895, 901, 1190, 1197
simpleelectronfraction (function), 1190, 1197
simplefailedaccretedmass (function), 895, 902
simplefailedaccretionrate (function), 895, 902
simplefailedfraction (function), 895, 902
simplemasstotal (function), 896, 902
simplesingleneutralheliumfraction (function), 1190, 1197
simplesingleneutralhydrogenfraction (function), 1191, 1197
simplesinglesinglyionizedheliumfraction (function), 1191, 1197
simpletemperature (function), 1191, 1197
simplevelocityscale (function), 896, 902
sine_integral (function), 1226
size_integer_scalar (function), 1937
sizefunction (type), 1118
sizefunctiondescriptor (type), 1118
sizefunctionwork (type), 1118
sneia_cumulative_number (function), 1824
sneia_cumulative_number_nagashima (function), 1824
sneia_cumulative_yield (function), 1824
sneia_cumulative_yield_nagashima (function), 1824
sneepopiii_cumulative_energy (function), 1822
sneepopiii_cumulative_energy_hegerwoosley (function), 1823
solve_for_radius (subroutine), 1076, 1077, 1081, 1082
sort (module), 1320
sort_do (interface), 1321
sort_do_double (subroutine), 1321
sort_do_double_c (subroutine), 1321
sort_do_integer (subroutine), 1321
sort_do_integer8 (subroutine), 1321
sort_do_integer8_both (subroutine), 1321
sort_do_integer8_c (subroutine), 1322
sort_do_integer_c (subroutine), 1322
sort_index_do (interface), 1322
sort_index_do_double (function), 1322
sort_index_do_double_c (subroutine), 1322
sort_index_do_integer8 (function), 1322
sort_index_do_integer8_c (subroutine), 1322
spectraltable (type), 1833
spectrapostprocessor (interface), 1837
spectrapostprocessorapplynull (subroutine), 1837
spectrapostprocessorclass (type), 1837
spectrapostprocessorconstructordefault (function), 1837
spectrapostprocessorconstructornamed (function), 1838
spectrapostprocessoridentity (interface), 1838, 1841
spectrapostprocessorinitialize (subroutine), 1838
spectrapostprocessorinoue2014 (interface), 1838, 1839
spectrapostprocessorisfinalizable (function), 1838
spectrapostprocessorlycsuppress (interface), 1838, 1840
spectrapostprocessormadau1995 (interface), 1838, 1840
spectrapostprocessormeiksin2006 (interface), 1838, 1841
spectrapostprocessorrecent (interface), 1838, 1842
spherical_collape_delta_critical_initialize (subroutine), 1879
spherical_collape_matter_lambda_delta_virial_tabulate (subroutine), 1879
spherical_collape_critical_overdensity (subroutine), 1879
spherical_collape_dark_energy_critical_overdensity (subroutine), 1878
spherical_collape_dark_energy_delta_critical_initialize (subroutine), 1878
spherical_collape_dark_energy_virial_density_contrast_tabulate (subroutine), 1878
spherical_collape_matter_dark_energy (module), 1877
spherical_collape_matter_dark_energy_state_retrieve (subroutine), 1878
spherical_collape_matter_dark_energy_state_store (subroutine), 1878
spherical_collape_matter_lambda (module), 1878
spherical_collape_matter_lambda_state_retrieve (subroutine), 1879
spherical_collape_matter_lambda_state_store (subroutine), 1879
spherical_shell_solid_angle_in_cylinder (function), 1148
sphericalcollapsematterdefaultconstructor (function), 1888, 1893
sphericalcollapsematterdefaultconstructor (function), 1888, 1894
sphericalcollapsematterlambdadestructor (function), 1889, 1894
sphericalcollapsematterlambdadestructor (function), 1889, 1894
sphericalcollapsematterlambdaretabulate (subroutine), 1889, 1894
sphericalcollapsematterlambdaretabulate (subroutine), 1889, 1894
sphericalcollapsematterlambdadestroyerstore (subroutine), 1889, 1894
sphericalcollapsematterlambdastatterestore (subroutine), 1889, 1894
sphericalcollapsematterlambdastatterestore (subroutine), 1889, 1894
spheroidabundancegas (function), 1643
spheroidabundancegasattributematch (function), 1644
spheroidabundancegasisgettable (function), 1644
spheroidabundancegasrate (subroutine), 1644
spheroidabundancestellar (function), 1644
spheroidabundancestellarattributematch (function), 1644
spheroidabundancestellarisgettable (function), 1644
spheroidabundancesstellarrate (subroutine), 1644
spheroidangularmomentum (function), 1644
spheroidangularmomentumattributematch (function), 1645
spheroidangularmomentumisgettable (function), 1645
spheroidangularmomentumrate (subroutine), 1645
spheroidcountlinked (function), 1645
spheroidcreatebyinterrupt (subroutine), 1645
spheroidcreatelinked (subroutine), 1645
spheroiddestroylinked (subroutine), 1645
spheroidenergygasinput (function), 1645
spheroidenergygasinputattributematch (function), 1646
spheroidenergygasinputisgettable (function), 1646
spheroidget (function), 1646
spheroidhalfmassradius (function), 1646
spheroidhalfmassradiusattributematch (function), 1646
spheroidhalfmassradiusisgettable (function), 1646
spheroidisinitialized (function), 1646
spheroidisinitializedattributematch (function), 1646
spheroidisinitializedisgettable (function), 1647
spheroidluminositiesstellar (function), 1647
spheroidluminositiesstellarattributematch (function), 1647
spheroidluminositiesstellarisgettable (function), 1647
spheroidmassgas (function), 1647
spheroidmassgasattributematch (function), 1647
spheroidmassgasisgettable (function), 1647
spheroidmassgassink (function), 1648
spheroidmassgassinkattributematch (function), 1648
spheroidmassgassinkisgettable (function), 1648
spheroidmassstellar (function), 1648
spheroidmassstellarattributematch (function), 1648
spheroidmassstellarisgettable (function), 1648
spheroidmassstellarrate (subroutine), 1648
spheroidnullbindingsabundances0inout (function), 1648
spheroidnullbindingdouble0inout (function), 1649
spheroidnullbindinghistory0inout (function), 1649
spheroidnullbindinginteger0in (function), 1649
spheroidnullbindinglogical0inout (function), 1649
spheroidnullbindingrateabundances0inout (subroutine), 1649
spheroidnullbindingratedouble0inout (subroutine), 1649
spheroidnullbindingratehistory0inout (subroutine), 1649
spheroidnullbindinginteger0in (subroutine), 1649
spheroidnullbindingrateinteger0inout (subroutine), 1649
spheroidnullbindingrateintegraluminosities0inout (subroutine), 1650
spheroidnullbindingsetabundances0inout (subroutine), 1650
spheroidnullbindingsetdouble0inout (subroutine), 1650
spheroidnullbindingsethistory0inout (subroutine), 1650
spheroidnullbindingsetinteger0in (subroutine), 1650
spheroidnullbindingsetlogical0inout (subroutine), 1650
spheroidnullbindingsetintegraluminosities0inout (subroutine), 1650
spheroidnullbindingsetstellarluminosities0inout (subroutine), 1650
spheroidnullbindingsetlogical0inout (subroutine), 1650
spheroidnullbindingsetstellarluminosities0inout (subroutine), 1650
spheroidnullbindingstellarluminosities0inout (function), 1650
spheroidradius (function), 1650
spheroidradiusattributematch (function), 1650
spheroidradiusisgettable (function), 1651
spheroidstandardabundancesgascount (function), 1651
spheroidstandardabundancesgasget (function), 1651
spheroidstandardabundancesgasrate (subroutine), 1651
spheroidstandardabundancesgasrategeneric (subroutine), 1651
spheroidstandardabundancesgasscale (subroutine), 1651
spheroidstandardabundancesgasset (subroutine), 1651
spheroidstandardabundancesstellarcount (function), 1651
spheroidstandardabundancesstellarget (function), 1652
spheroidstandardabundancesstellarrate (subroutine), 1652
spheroidstandardabundancesstellarrategeneric (subroutine), 1652
spheroidstandardabundancesstellarscale (subroutine), 1652
spheroidstandardabundancesstellarset (subroutine), 1652
spheroidstandardangularmomentumcount (function), 1652
spheroidstandardangularmomentumget (function), 1652
spheroidstandardangularmomentumrate (subroutine), 1652
spheroidstandardangularmomentumrategeneric (subroutine), 1652
spheroidstandardangularmomentumscale (subroutine), 1653
spheroidstandardcreatefunctionset (subroutine), 1653
spheroidstandardenergygasinputrate (subroutine), 1653
spheroidstandardenergygasinputratefunction (subroutine), 1653
spheroidstandardenergygasinputrateisattached (function), 1653
spheroidstandardisinitializedget (function), 1653
spheroidstandardisinitializedset (subroutine), 1653
spheroidstandardluminositiesstellarcount (function), 1654
spheroidstandardluminositiesstellarget (function), 1654
spheroidstandardluminositiesstellarrate (subroutine), 1654
spheroidstandardluminositiesstellarrategeneric (subroutine), 1654
spheroidstandardluminositiesstellarscale (subroutine), 1654
spheroidstandardluminositiesstellarset (subroutine), 1654
spheroidstandardmassgascount (function), 1654
spheroidstandardmassgasget (function), 1654
spheroidstandardmassgasrate (subroutine), 1655
spheroidstandardmassgasscale (subroutine), 1655
spheroidstandardmassgasset (subroutine), 1655
spheroidstandardmasssinkrate (subroutine), 1655
spheroidstandardmasssinkratefunction (subroutine), 1655
spheroidstandardmasssinkrateisattached (function), 1655
spheroidstandardmassstellarcount (function), 1655
spheroidstandardmassstellarget (function), 1656
spheroidstandardmassstellarrate (subroutine), 1656
spheroidstandardmassstellarrategeneric (subroutine), 1656
spheroidstandardmassstellarscale (subroutine), 1656
spheroidstandardmassstellarset (subroutine), 1656
spheroidstandardradiusget (function), 1656
spheroidstandardradiusset (subroutine), 1656
spheroidstandardstarformationhistorycount (function), 1656
spheroidstandardstarformationhistoryget (function), 1656
spheroidstandardstarformationhistoryrate (subroutine), 1657
spheroidstandardstarformationhistoryrateregeneric (subroutine), 1657
spheroidstandardstarformationhistoryscale (subroutine), 1657
spheroidstandardstarformationhistoryset (subroutine), 1657
spheroidstandardstarformationrateget (function), 1657
spheroidstandardstarformationrategetfunction (subroutine), 1657
spheroidstandardstarformationrategetisattached (function), 1657
spheroidstandardstellarpropertieshistorycount (function), 1657
spheroidstandardstellarpropertieshistoryget (function), 1658
spheroidstandardstellarpropertieshistoryrate (subroutine), 1658
spheroidstandardstellarpropertieshistoryrateregeneric (subroutine), 1658
spheroidstandardstellarpropertieshistoryscale (subroutine), 1658
spheroidstandardstellarpropertieshistoryset (subroutine), 1658
spheroidstandardvelocityget (function), 1658
spheroidstandardvelocityset (subroutine), 1658
spheroidstarformationhistory (function), 1658
spheroidstarformationhistoryattributematch (function), 1659
spheroidstarformationhistoryisgettable (function), 1659
spheroidstarformationhistoryrate (subroutine), 1659
spheroidstarformationrate (function), 1659
spheroidstarformationrateattributematch (function), 1659
spheroidstarformationrateisgettable (function), 1659
spheroidstellarpropertieshistory (function), 1659
spheroidstellarpropertieshistoryattributematch (function), 1659
spheroidstellarpropertieshistoryisgettable (function), 1659
spheroidstellarpropertieshistoryrate (subroutine), 1660
spheroidvelocity (function), 1660
spheroidvelocityattributematch (function), 1660
spheroidvelocityisgettable (function), 1660
spincountlinked (function), 1660
spincreatebyinterrupt (subroutine), 1660
spinctrelinked (subroutine), 1660
spindestroylinked (subroutine), 1661
spinget (function), 1661
spinnullbindingdouble0inout (function), 1661
spinnullbindingdouble1inout (function), 1661
spinnullbindinginteger0in (function), 1661
spinnullbindingratedouble0inout (subroutine), 1661
spinnullbindingratedouble1inout (subroutine), 1661
spinnullbindinginteger0in (subroutine), 1661
spinnullbindingsetdouble0inout (subroutine), 1661
spinnullbindingsetdouble1inout (subroutine), 1661
spinnullbindingsetinteger0in (subroutine), 1662
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>spinpreset3dspinvectorcount</td>
<td>(function)</td>
<td>1662</td>
</tr>
<tr>
<td>spinpreset3dspinvectorget</td>
<td>(function)</td>
<td>1662</td>
</tr>
<tr>
<td>spinpreset3dspinvectorgrowthrateget</td>
<td>(function)</td>
<td>1662</td>
</tr>
<tr>
<td>spinpreset3dspinvectorgrowthrateset</td>
<td>(subroutine)</td>
<td>1662</td>
</tr>
<tr>
<td>spinpreset3dspinvectorrate</td>
<td>(subroutine)</td>
<td>1662</td>
</tr>
<tr>
<td>spinpreset3dspinvectorscale</td>
<td>(subroutine)</td>
<td>1662</td>
</tr>
<tr>
<td>spinpreset3dspinvectorset</td>
<td>(subroutine)</td>
<td>1662</td>
</tr>
<tr>
<td>spinpresetspincount</td>
<td>(function)</td>
<td>1662</td>
</tr>
<tr>
<td>spinpresetspinget</td>
<td>(function)</td>
<td>1663</td>
</tr>
<tr>
<td>spinpresetspingetgrowthrate</td>
<td>(function)</td>
<td>1663</td>
</tr>
<tr>
<td>spinpresetspingetgrowthrateset</td>
<td>(subroutine)</td>
<td>1663</td>
</tr>
<tr>
<td>spinpresetspinrate</td>
<td>(subroutine)</td>
<td>1663</td>
</tr>
<tr>
<td>spinpresetspinscale</td>
<td>(subroutine)</td>
<td>1663</td>
</tr>
<tr>
<td>spinrandomspincount</td>
<td>(function)</td>
<td>1663</td>
</tr>
<tr>
<td>spinrandomspinget</td>
<td>(function)</td>
<td>1663</td>
</tr>
<tr>
<td>spinrandomspingrowthrate</td>
<td>(function)</td>
<td>1664</td>
</tr>
<tr>
<td>spinrandomspinrate</td>
<td>(subroutine)</td>
<td>1664</td>
</tr>
<tr>
<td>spinrandomspinscale</td>
<td>(subroutine)</td>
<td>1664</td>
</tr>
<tr>
<td>spinrandomspinset</td>
<td>(subroutine)</td>
<td>1664</td>
</tr>
<tr>
<td>spinspin</td>
<td>(function)</td>
<td>1664</td>
</tr>
<tr>
<td>spinspinattributematch</td>
<td>(function)</td>
<td>1664</td>
</tr>
<tr>
<td>spinpingrowthrate</td>
<td>(function)</td>
<td>1664</td>
</tr>
<tr>
<td>spinpingrowthrateattributematch</td>
<td>(function)</td>
<td>1664</td>
</tr>
<tr>
<td>spinpingrowthrateisgettable</td>
<td>(function)</td>
<td>1664</td>
</tr>
<tr>
<td>spinpinisgettable</td>
<td>(function)</td>
<td>1665</td>
</tr>
<tr>
<td>spinpinvector</td>
<td>(function)</td>
<td>1665</td>
</tr>
<tr>
<td>spinpinvectorattributematch</td>
<td>(function)</td>
<td>1665</td>
</tr>
<tr>
<td>spinpinvectorgrowthrate</td>
<td>(function)</td>
<td>1665</td>
</tr>
<tr>
<td>spinpinvectorgrowthrateattributematch</td>
<td>(function)</td>
<td>1665</td>
</tr>
<tr>
<td>spinpinvectorisgettable</td>
<td>(function)</td>
<td>1665</td>
</tr>
<tr>
<td>split</td>
<td>(interface)</td>
<td>1224</td>
</tr>
<tr>
<td>split_ch</td>
<td>(subroutine)</td>
<td>1224</td>
</tr>
<tr>
<td>split_vs</td>
<td>(subroutine)</td>
<td>1224</td>
</tr>
<tr>
<td>spmpar</td>
<td>(function)</td>
<td>947</td>
</tr>
<tr>
<td>star_ejected_mass</td>
<td>(function)</td>
<td>1819</td>
</tr>
<tr>
<td>star_ejected_mass_file</td>
<td>(function)</td>
<td>1822</td>
</tr>
<tr>
<td>star_formation.feedback.disks.Creasey2012.F90</td>
<td>(file)</td>
<td>1800</td>
</tr>
<tr>
<td>star_formation.feedback.disks.F90</td>
<td>(file)</td>
<td>1802</td>
</tr>
<tr>
<td>star_formation.feedback.disks.fixed.F90</td>
<td>(file)</td>
<td>1803</td>
</tr>
<tr>
<td>star_formation.feedback.disks.halo_scaling.F90</td>
<td>(file)</td>
<td>1803</td>
</tr>
<tr>
<td>star_formation.feedback.disks.power_law.F90</td>
<td>(file)</td>
<td>1804</td>
</tr>
<tr>
<td>star_formation.feedback.spheroids.F90</td>
<td>(file)</td>
<td>1804</td>
</tr>
<tr>
<td>star_formation.feedback_expulsion.disks.F90</td>
<td>(file)</td>
<td>1805</td>
</tr>
<tr>
<td>star_formation.feedback_expulsion.disks.null.F90</td>
<td>(file)</td>
<td>1806</td>
</tr>
<tr>
<td>star_formation.feedback_expulsion.disks.superwind.F90</td>
<td>(file)</td>
<td>1807</td>
</tr>
<tr>
<td>star_formation.feedback_expulsion.spheroids.F90</td>
<td>(file)</td>
<td>1807</td>
</tr>
</tbody>
</table>
star_formation.feedback_expulsion.spheroids.null.F90 (file), 1808
star_formation.feedback_expulsion.spheroids.superwind.F90 (file), 1808
star_formation.IMF.Baugh2005.TopHeavy.F90 (file), 1783
star_formation.IMF.Chabrier.F90 (file), 1785
star_formation.IMF.F90 (file), 1786
star_formation.IMF.Kennicutt.F90 (file), 1790
star_formation.IMF.Kroupa.F90 (file), 1791
star_formation.IMF.Miller-Scalo.F90 (file), 1793
star_formation.IMF.piecewise_power_law.F90 (file), 1796
star_formation.IMF.Salpeter.F90 (file), 1794
star_formation.IMF.Scalo.F90 (file), 1795
star_formation.IMF.select.disksSpheroid.F90 (file), 1798
star_formation.IMF.select.fixed.F90 (file), 1798
star_formation.IMF.utilities.F90 (file), 1799
star_formation.IMF.utilities.piecewise_power_laws.F90 (file), 1799
star_formation.rate_surface_density.disks.Blitz-Rosolowsky.F90 (file), 1809
star_formation.rate_surface_density.disks.extended_Schmidt.F90 (file), 1812
star_formation.rate_surface_density.disks.F90 (file), 1810
star_formation.rate_surface_density.disks.Kennicutt-Schmidt.F90 (file), 1811
star_formation.rate_surface_density.disks.KMT09.F90 (file), 1810
star_formation.timescales.disks.Baugh2005.cpp (file), 1813
star_formation.timescales.disks.dynamical_time.F90 (file), 1814
star_formation.timescales.disks.F90 (file), 1813
star_formation.timescales.disks.fixed.F90 (file), 1814
star_formation.timescales.disks.halo_scaling.F90 (file), 1815
star_formation.timescales.disks.integrated_surface_density.F90 (file), 1816
star_formation.timescales.spheroids.dynamical_time.F90 (file), 1817
star_formation.timescales.spheroids.F90 (file), 1816
star_formation_expulsive_feedback_disk_outflow_rate (function), 1806
star_formation_expulsive_feedback_disk_outflow_rate_null (function), 1806
star_formation_expulsive_feedback_disk_outflow_rate_sw (function), 1807
star_formation_expulsive_feedback_disks.initialize (subroutine), 1806
star_formation_expulsive_feedback_disks_null (module), 1806
star_formation_expulsive_feedback_disks_null.initialize (subroutine), 1806
star_formation_expulsive_feedback_disks_superwind (module), 1807
star_formation_expulsive_feedback_disks_sw.initialize (subroutine), 1807
star_formation_expulsive_feedback_spheroids.initialize (subroutine), 1808
star_formation_expulsive_feedback_spheroids_outflow_rate (function), 1808
star_formation_expulsive_feedback_spheroids_outflow_rate_null (function), 1808
star_formation_expulsive_feedback_spheroids_outflow_rate_sw (function), 1809
star_formation_expulsive_feedback_spheroids.initialize (subroutine), 1808
star_formation_expulsive_feedback_spheroids_null (module), 1808
star_formation_expulsive_feedback_spheroids_null.initialize (subroutine), 1808
star_formation_expulsive_feedback_spheroids_superwind (module), 1808
star_formation_expulsive_feedback_spheroids_sw.initialize (subroutine), 1809
star_formation_feedback_disk_outflow_rate (function), 1802
star_formation_feedback_disk_outflow_rate_creasey2012 (function), 1800
star_formation_feedback_disk_outflow_rate_creasey2012_integrand (function), 1801
star_formation_feedback_disk_outflow_rate_fixed (function), 1803
star_formation_feedback_disk_outflow_rate_halo_scaling (function), 1803
star_formation_feedback_disk_outflow_rate_power_law (function), 1804
star_formation_feedback_disk_outflow_rate_template (function), 1802
star_formation_feedback_disk_outflow_rate_template (interface), 1802
star_formation_feedback_disks (module), 1802
star_formation_feedback_disks_creasey2012 (module), 1800
star_formation_feedback_disks_creasey2012_initialize (subroutine), 1802
star_formation_feedback_disks_fixed (module), 1803
star_formation_feedback_disks_fixed_initialize (subroutine), 1803
star_formation_feedback_disks_halo_scaling (module), 1803
star_formation_feedback_disks_halo_scaling_initialize (subroutine), 1803
star_formation_feedback_disks_initialize (subroutine), 1802
star_formation_feedback_disks_power_law (module), 1804
star_formation_feedback_expulsion_disks (module), 1806
star_formation_feedback_expulsion_spheroids (module), 1807
star_formation_feedback_spheroid_outflow_rate (function), 1805
star_formation_feedback_spheroid_outflow_rate_power_law (function), 1805
star_formation_feedback_spheroid_outflow_rate_template (function), 1805
star_formation_feedback_spheroid_outflow_rate_template (interface), 1805
star_formation_feedback_spheroids (module), 1804
star_formation_feedback_spheroids_initialize (subroutine), 1805
star_formation_feedback_spheroids_power_law (module), 1805
star_formation_feedback_spheroids_power_law_initialize (subroutine), 1805
star_formation_histories_metallicity_split (module), 1142
star_formation_histories_metallicity_split_initialize (subroutine), 1142
star_formation_histories_null (module), 1143
star_formation_histories_null_initialize (subroutine), 1143
star_formation_history_create (subroutine), 1141
star_formation_history_create_metallicity_split (subroutine), 1142
star_formation_history_create_null (subroutine), 1143
star_formation_history_metalllicity_split_make_history (subroutine), 1142
star_formation_history_output (subroutine), 1141
star_formation_history_output_metallicity_split (subroutine), 1142
star_formation_history_output_null (subroutine), 1143
star_formation_history_record (subroutine), 1141
star_formation_history_record_metallicity_split (subroutine), 1142
star_formation_history_record_null (subroutine), 1143
star_formation_history_scales (subroutine), 1141
star_formation_history_scales_metallicity_split (subroutine), 1143
star_formation_history_scales_null (subroutine), 1144
star_formation_imf (module), 1786
star_formation_imf_baugh2005topheavy (module), 1784
star_formation_imf_chabrier (module), 1785
star_formation_imf_initialize (subroutine), 1790
star_formation_imf_initialize_baugh2005topheavy (subroutine), 1784
star_formation_imf_initialize_chabrier (subroutine), 1785
star_formation_imf_initialize_kennicutt (subroutine), 1790
star_formation_imf_initialize_kroupa (subroutine), 1792
star_formation_imf_initialize_millerscalo (subroutine), 1793
star_formation_imf_initialize_piecewisepowerlaw (subroutine), 1797
star_formation_imf_initialize_salpeter (subroutine), 1794
star_formation_imf_initialize_scalo (subroutine), 1795
star_formation_imf_kennicutt (module), 1790
star_formation_imf_kroupa (module), 1791
star_formation_imf_label (function), 1952
star_formation_imf_maximum_mass_baugh2005topheavy (subroutine), 1784
star_formation_imf_maximum_mass_chabrier (subroutine), 1785
star_formation_imf_maximum_mass_kennicutt (subroutine), 1790
star_formation_imf_maximum_mass_kroupa (subroutine), 1792
star_formation_imf_maximum_mass_millerscalo (subroutine), 1793
star_formation_imf_maximum_mass_piecewisepowerlaw (subroutine), 1797
star_formation_imf_maximum_mass_salpeter (subroutine), 1794
star_formation_imf_maximum_mass_scalo (subroutine), 1795
star_formation_imf_millerscalo (module), 1793
star_formation_imf_minimum_mass_baugh2005topheavy (subroutine), 1784
star_formation_imf_minimum_mass_chabrier (subroutine), 1785
star_formation_imf_minimum_mass_kennicutt (subroutine), 1791
star_formation_imf_minimum_mass_kroupa (subroutine), 1792
star_formation_imf_minimum_mass_millerscalo (subroutine), 1793
star_formation_imf_minimum_mass_piecewisepowerlaw (subroutine), 1797
star_formation_imf_minimum_mass_salpeter (subroutine), 1794
star_formation_imf_minimum_mass_scalo (subroutine), 1796
star_formation_imf_phi_baugh2005topheavy (subroutine), 1784
star_formation_imf_phi_chabrier (subroutine), 1785
star_formation_imf_phi_kennicutt (subroutine), 1791
star_formation_imf_phi_kroupa (subroutine), 1792
star_formation_imf_phi_millerscalo (subroutine), 1793
star_formation_imf_phi_piecewisepowerlaw (subroutine), 1797
star_formation_imf_phi_salpeter (subroutine), 1794
star_formation_imf_phi_scalo (subroutine), 1796
star_formation_imf_piecewisepowerlaw (module), 1796
star_formation_imf_ppl (module), 1799
star_formation_imf_recycled_instantaneous_baugh2005topheavy (subroutine), 1784
star_formation_imf_recycled_instantaneous_chabrier (subroutine), 1785
star_formation_imf_recycled_instantaneous_kennicutt (subroutine), 1791
star_formation_imf_recycled_instantaneous_kroupa (subroutine), 1792
star_formation_imf_recycled_instantaneous_millerscalo (subroutine), 1793
star_formation_imf_recycled_instantaneous_piecewisepowerlaw (subroutine), 1797
star_formation_imf_recycled_instantaneous_salpeter (subroutine), 1795
star_formation_imf_recycled_instantaneous_scalo (subroutine), 1796
star_formation_imf_register_baugh2005topheavy (subroutine), 1784
star_formation_imf_register_chabrier (subroutine), 1786
star_formation_imf_register_kennicutt (subroutine), 1791
star_formation_imf_register_kroupa (subroutine), 1792
star_formation_imf_register_millerscalo (subroutine), 1793
star_formation_imf_register_name_baugh2005topheavy (subroutine), 1784
star_formation_imf_register_name_chabrier (subroutine), 1786
star_formation_imf_register_name_kennicutt (subroutine), 1791
star_formation_imf_register_name_kroupa (subroutine), 1792
star_formation_imf_register_name_millerscalo (subroutine), 1794
star_formation_imf_register_name_piecewisepowerlaw (subroutine), 1797
star_formation_imf_register_name_salpeter (subroutine), 1795
star_formation_imf_register_name_scalo (subroutine), 1796
star_formation_imf_register_piecewisepowerlaw (subroutine), 1797
star_formation_imf_register_salpeter (subroutine), 1795
star_formation_imf_register_scalo (subroutine), 1796
star_formation_imf_salpeter (module), 1794
star_formation_imf_scalo (module), 1795
star_formation_imf_select_disk_spheroid (module), 1798
star_formation_imf_select_fixed (module), 1798
star_formation_imf_tabulate_baugh2005topheavy (subroutine), 1784
star_formation_imf_tabulate_chabrier (subroutine), 1786
star_formation_imf_tabulate_kennicutt (subroutine), 1791
star_formation_imf_tabulate_kroupa (subroutine), 1792
star_formation_imf_tabulate_millerscalo (subroutine), 1794
star_formation_imf_tabulate_piecewisepowerlaw (subroutine), 1797
star_formation_imf_tabulate_salpeter (subroutine), 1795
star_formation_imf_tabulate_scalo (subroutine), 1796
star_formation_imf_utilities (module), 1799
star_formation_imf_yield_instantaneous_baugh2005topheavy (subroutine), 1785
star_formation_imf_yield_instantaneous_chabrier (subroutine), 1786
star_formation_imf_yield_instantaneous_kennicutt (subroutine), 1791
star_formation_imf_yield_instantaneous_kroupa (subroutine), 1793
star_formation_imf_yield_instantaneous_millerscalo (subroutine), 1794
star_formation_imf_yield_instantaneous_piecewisepowerlaw (subroutine), 1797
star_formation_imf_yield_instantaneous_salpeter (subroutine), 1795
star_formation_imf_yield_instantaneous_scalo (subroutine), 1796
star_formation_rate_integrand_surface_density (function), 1816
star_formation_rate_surface_density_disk (function), 1810
star_formation_rate_surface_density_disk_br (function), 1809
star_formation_rate_surface_density_disk_exschmidt (function), 1813
star_formation_rate_surface_density_disk_kmt09 (function), 1811
star_formation_rate_surface_density_disk_ks (function), 1812
star_formation_rate_surface_density_disks (module), 1810
star_formation_rate_surface_density_disks_br (module), 1809
star_formation_rate_surface_density_disks_br_initialize (subroutine), 1809
star_formation_rate_surface_density_disks_br_reset (subroutine), 1810
star_formation_rate_surface_density_disks_exschmidt (module), 1812
star_formation_rate_surface_density_disks_exschmidt_initialize (subroutine), 1813
star_formation_rate_surface_density_disks_exschmidt_reset (subroutine), 1813
star_formation_rate_surface_density_disks_initialize (subroutine), 1810
star_formation_rate_surface_density_disks_kmt09 (module), 1810
star_formation_rate_surface_density_disks_kmt09_initialize (subroutine), 1811
star_formation_rate_surface_density_disks_kmt09_reset (subroutine), 1811
star_formation_rate_surface_density_disks_ks (module), 1811
star_formation_rate_surface_density_disks_ks_initialize (subroutine), 1812
star_formation_rate_surface_density_disks_ks_reset (subroutine), 1812
star_formation_timescale_disk (function), 1813
star_formation_timescale_disk_dynamical_time (function), 1814
star_formation_timescale_disk_fixed (function), 1815
star_formation_timescale_disk_halo_scaling (function), 1815
star_formation_timescale_disk_integrated_sd (function), 1816
star_formation_timescale_disks_dynamical_time (module), 1814
star_formation_timescale_disks_dynamical_time_initialize (subroutine), 1814
star_formation_timescale_disks_fixed (module), 1815
star_formation_timescale_disks_halo_scaling (module), 1815
star_formation_timescale_disks_halo_scaling_initialize (subroutine), 1815
star_formation_timescale_disks_halo_scaling_reset (subroutine), 1816
star_formation_timescale_disks_initialize (subroutine), 1814
star_formation_timescale_disks_integrated_sd (module), 1816
star_formation_timescale_disks_integrated_sd_initialize (subroutine), 1816
star_formation_timescale_spheroid (function), 1817
star_formation_timescale_spheroid_dynamical_time (function), 1817
star_formation_timescale_spheroids_dynamical_time (module), 1817
star_formation_timescale_spheroids_dynamical_time_initialize (subroutine), 1817
star_formation_timescale_spheroids_initialize (subroutine), 1817
star_formation_timescales_disks (module), 1813
star_formation_timescales_spheroids (module), 1816
star_initial_mass (function), 1819
star_initial_mass_file (function), 1822
star_is_evolved (function), 1790
star_lifetime (function), 1820
star_lifetime_file (function), 1822
star_metal_yield_mass (function), 1820
star_metal_yield_mass_file (function), 1822
state_initialize (subroutine), 1153
stateful_types (module), 1976
statefuldouble (type), 1977
statefullinteger (type), 1977
statefullogical (type), 1977
statistics.mass_function.covariance.F90 (file), 1818
statistics_mass_function_covariance (module), 1818
stellar_astrophysics (module), 1819
stellar_astrophysics.F90 (file), 1819
stellar_astrophysics.feedback.F90 (file), 1820
stellar_astrophysics.feedback.standard.F90 (file), 1821
stellar_astrophysics.file.F90 (file), 1821
stellar_astrophysics.supernovae_PopulationIII.F90 (file), 1822
stellar_astrophysics.supernovae_PopulationIII.HegerWoosley.F90 (file), 1823
stellar_astrophysics.supernovae_type_Ia.F90 (file), 1823
stellar_astrophysics.supernovae_type_Ia.Nagashima.F90 (file), 1824
stellar_astrophysics.tracks.F90 (file), 1825
stellar_astrophysics.tracks.file.F90 (file), 1825
stellar_astrophysics.winds.F90 (file), 1826
stellar_astrophysics.winds.Leitherer1992.F90 (file), 1827
CODE INDEX

stellar_populations.spectra.Conroy_et_al.F90 (file), 1831
stellar_populations.spectra.F90 (file), 1832
stellar_populations.spectra.file.F90 (file), 1833
stellar_populations.spectra.postprocess.F90 (file), 1834
stellar_populations.spectra.postprocess.identity.F90 (file), 1841
stellar_populations.spectra.postprocess.Inoue2014.F90 (file), 1839
stellar_populations.spectra.postprocess.Lyman_continuum_suppress.F90 (file), 1839
stellar_populations.spectra.postprocess.Madau1995.F90 (file), 1840
stellar_populations.spectra.postprocess.Meiksin2006.F90 (file), 1840
stellar_populations.spectra.postprocess.recent.F90 (file), 1841
stellar_tracks_initialize (subroutine), 1825
stellar_tracks_initialization_file (subroutine), 1826
stellar_tracks_interpolation_do (function), 1826
stellar_tracks_interpolation_get (subroutine), 1826
stellar_winds_initialize (subroutine), 1826
stellar_winds_leitherer1992_initialize (subroutine), 1827
stellar_winds_mass_loss_rate (function), 1827
stellar_winds_mass_loss_rate_leitherer1992 (function), 1827
stellar_winds_terminal_velocity (function), 1827
stellar_winds_terminal_velocity_leitherer1992 (function), 1827
stellarluminosities (type), 1729
stellarluminositiesmax (function), 1729
stellarspectraconvolution (function), 1744
store_unit_attributes_galacticus (subroutine), 1360
store_unit_attributes_irate (subroutine), 1360
string_c_to_fortran (function), 1981
string_count_words (function), 1982
string_handling (module), 1977
string_join (function), 1982
string_levenshstein_distance (function), 1982
string_lower_case (function), 1982
string_split_words (interface), 1982
string_split_words_char (subroutine), 1982
string_split_words_varstring (subroutine), 1982
string_strip (function), 1982
string_subscript (function), 1982
string_superscript (function), 1982
string_upper_case (function), 1983
string_upper_case_first (function), 1983
structure_formation.cosmological_mass_variance.filtered_power_spectrum.F90 (file), 1842
structure_formation.critical_overdensity.F90 (file), 1843
structure_formation.critical_overdensity.Kitayama_Suto1996.F90 (file), 1845
structure_formation.critical_overdensity.mass_scaling.null.F90 (file), 1846
structure_formation.critical_overdensity.mass_scaling.warm_dark_matter.F90 (file), 1846
structure_formation.excursion_sets.barrier.critical_overdensity.F90 (file), 1849
structure_formation.excursion_sets.barrier.F90 (file), 1848
structure_formation.excursion_sets.barrier.linear.F90 (file), 1850
structure_formation.excursion_sets.barrier.quadratic.F90 (file), 1850
structure_formation.excursion_sets.barrier.remap.null.F90 (file), 1851
structure_formation.excursion_sets.barrier.remap.scale.F90 (file), 1852
structure_formation.excursion_sets.barrier.remap.Sheth-Mo-Tormen.F90 (file), 1851
structure_formation.excursion_sets.first_crossing_distribution.F90 (file), 1853
structure_formation.excursion_sets.first_crossing_distribution.Farahi.F90 (file), 1855
structure_formation.excursion_sets.first_crossing_distribution.linear_barrier.F90 (file), 1858
structure_formation.excursion_sets.first_crossing_distribution.Zhang_Hui.F90 (file), 1855
structure_formation.excursion_sets.first_crossing_distribution.Zhang_Hui.utilities.F90 (file), 1856
structure_formation.excursion_sets.first_crossing_distribution.Zhang_Hui_high_order.F90 (file), 1857
structure_formation.halo_bias.F90 (file), 1860
structure_formation.halo_bias.Press-Schechter.F90 (file), 1861
structure_formation.halo_bias.SMT.F90 (file), 1862
structure_formation.halo_mass_function.F90 (file), 1862
structure_formation.halo_mass_function.modifier.simple_systematic.F90 (file), 1865
structure_formation.halo_mass_function.Press-Schechter.F90 (file), 1863
structure_formation.halo_mass_function.Sheth-Tormen.F90 (file), 1864
structure_formation.halo_mass_function.Tinker2008.F90 (file), 1864
structure_formation.linear_growth.F90 (file), 1865
structure_formation.linear_growth.simple.F90 (file), 1867
structure_formation.power_spectrum.F90 (file), 1868
structure_formation.power_spectrum.nonlinear.CosmicEmu.F90 (file), 1870
structure_formation.power_spectrum.nonlinear.F90 (file), 1870
structure_formation.power_spectrum.nonlinear.linear.F90 (file), 1871
structure_formation.power_spectrum.nonlinear.PeacockDodds1996.F90 (file), 1871
structure_formation.power_spectrum.primordial.F90 (file), 1872
structure_formation.power_spectrum.primordial.power_law.F90 (file), 1873
structure_formation.power_spectrum.primordial.transferred.F90 (file), 1873
structure_formation.power_spectrum.variance.window_function.F90 (file), 1874
structure_formation.power_spectrum.variance.window_function.sharp_kSpace.F90 (file), 1874
structure_formation.power_spectrum.variance.window_function.top_hat.F90 (file), 1875
structure_formation.power_spectrum.variance.window_function.top_hat_kSpace.sharp_hybrid.F90 (file), 1876
structure_formation.spherical_collapse.matter_dark_energy.F90 (file), 1877
structure_formation.spherical_collapse.matter_lambda.F90 (file), 1878
structure_formation.transfer_function.BBKS.F90 (file), 1880
structure_formation.transfer_function.CAMB.F90 (file), 1881
structure_formation.transfer_function.Eisenstein_Hu.F90 (file), 1881
structure_formation.transfer_function.F90 (file), 1882
structure_formation.transfer_function.file.F90 (file), 1883
structure_formation.transfer_function.null.F90 (file), 1884
structure_formation.virial_density_contrast.Bryan_Norman.F90 (file), 1885
structure_formation.virial_density_contrast.F90 (file), 1885
structure_formation.virial_density_contrast.fixed.F90 (file), 1892
structure_formation.virial_density_contrast.friends-of-friends.F90 (file), 1892
structure_formation.virial_density_contrast.Kitayama_Suto1996.F90 (file), 1891
structure_formation.virial_density_contrast.sphericalCollapse_matter_dark_energy.F90 (file), 1893
structure_formation.virial_density_contrast.sphericalCollapse_matter_lambda.F90 (file), 1894
subresolution_fraction_integrand_generalized (function), 1234
supernovae_population_iii (module), 1822
supernovae_population_iii_hegerwoosley (module), 1823
supernovae_population_iii_hegerwoosley_initialize (subroutine), 1823
supernovae_population_iii_initialize (subroutine), 1823
supernovae_type_ia (module), 1823
supernovae_type_ia_initialize (subroutine), 1824
supernovae_type_ia_nagashima (module), 1824
supernovae_type_ia_nagashima_initialize (subroutine), 1824
sussingangularmomenta3davailable (function), 1259, 1264
sussingangularmomentaavailable (function), 1259, 1264
sussingangularmomentaincludesubhalos (function), 1260, 1264
sussingclose (subroutine), 1260, 1264
sussingcubelength (function), 1260, 1264
sussingdefaultconstructor (function), 1260, 1264
sussingdestructor (subroutine), 1260, 1264
sussingimport (subroutine), 1260, 1265
sussinginsubvolume (function), 1260, 1265
sussinginsubvolumeid (function), 1260, 1265
sussingmassesincludesubhalos (function), 1261, 1265
sussingnodecount (function), 1261, 1265
sussingopen (subroutine), 1261, 1265
sussingparticlecountavailable (function), 1261, 1265
sussingperiodicseparation (function), 1261, 1266
sussingpositionsareperiodic (function), 1261, 1266
sussingpositionsavailable (function), 1261, 1266
sussingreadhaloascii (subroutine), 1261, 1266
sussingscaleradiiavailable (function), 1262, 1266
sussingspin3davailable (function), 1262, 1266
sussingspinavailable (function), 1262, 1266
sussingsubhalotrace (subroutine), 1262, 1266
sussingsubhalotracecount (function), 1262, 1266
sussingtreecount (function), 1262, 1267
sussingtreeindex (function), 1262, 1267
sussingtreeindicesread (subroutine), 1262, 1267
sussingtreesareselfcontained (function), 1263, 1267
sussingtreeshavesubhalos (function), 1263, 1267
sussingtreeweight (function), 1263, 1267
sussingvaluesisbad (function), 1263, 1267
sussingvelocitiesincludehubbleflow (function), 1263, 1267
sussingvelocitydispersionavailable (function), 1263, 1268
sussingvelocitymaximumavailable (function), 1263, 1268
system.command.F90 (file), 1895
system.load.F90 (file), 1895
system_command (module), 1895
system_command_do (subroutine), 1895
system_load (module), 1895
system_load_get (subroutine), 1895
system_processor_count (function), 1896

table (type), 1730
tableid (type), 1730
tableid_find_effective_x (function), 1730
tableid_is_monotonic (function), 1730
tableid_size (function), 1731
tableid_x (function), 1731
tableid_xs (function), 1731
tableid_y (function), 1731
tableid_ys (function), 1731
tabledgeneric (type), 1731
tabledlinearcspline (type), 1731
tabledlinearlinear (type), 1731
tabledlogarithmiccspline (type), 1731
tabledlogarithmiclinear (type), 1732
table_id_destroy (subroutine), 1732
table_id_reverse (subroutine), 1732
table_generic_id_create (subroutine), 1732
table_generic_id_destroy (subroutine), 1732
table_generic_id_interpolate (function), 1732
table_generic_id_interpolate_gradient (function), 1732
table_generic_id_populate (subroutine), 1732
table_generic_id_populate_single (subroutine), 1733
table_linear_id_create (subroutine), 1733
table_linear_id_interpolate (function), 1733
table_linear_id_interpolate_gradient (function), 1733
table_linear_id_populate (subroutine), 1733
table_linear_id_populate_single (subroutine), 1733
table_linear_cspline_id_coefficients (subroutine), 1733
table_linear_cspline_id_compute_spline (subroutine), 1733
table_linear_cspline_id_create (subroutine), 1733
table_linear_cspline_id_destroy (subroutine), 1734
table_linear_cspline_id_interpolate (function), 1734
table_linear_cspline_id_interpolate_gradient (function), 1734
table_linear_cspline_id_populate (subroutine), 1734
table_linear_cspline_id_populate_single (subroutine), 1734
table_logarithmic_id_create (subroutine), 1734
table_logarithmic_id_interpolate (function), 1734
table_logarithmic_id_interpolate_gradient (function), 1734
table_logarithmic_id_x (function), 1734
table_logarithmic_id_xs (function), 1735
table_logarithmic_cspline_id_create (subroutine), 1735
table_logarithmic_cspline_id_interpolate (function), 1735
table_logarithmic_cspline_id_interpolate_gradient (function), 1735
table_logarithmic_cspline_id_x (function), 1735
table_logarithmic_cspline_id_xs (function), 1735
tables (module), 1729
tcollapse (function), 1880
temperaturecmbsimple (function), 1002, 1003
CODE INDEX

tensor (type), 1736
tensor_r2_d3_sym_add (function), 1736
tensor_r2_d3_sym_assign_from (subroutine), 1736
tensor_r2_d3_sym_assign_to (subroutine), 1736
tensor_r2_d3_sym_builder (subroutine), 1736
tensor_r2_d3_sym_contract (function), 1736
tensor_r2_d3_sym_deserialize (subroutine), 1736
tensor_r2_d3_sym_destroy (subroutine), 1736
tensor_r2_d3_sym_double_contract (function), 1737
tensor_r2_d3_sym_dump (subroutine), 1737
tensor_r2_d3_sym_dump_raw (subroutine), 1737
tensor_r2_d3_sym_from_matrix (subroutine), 1737
tensor_r2_d3_sym_increment (subroutine), 1737
tensor_r2_d3_sym_is_zero (function), 1737
tensor_r2_d3_sym_matrix_equality (function), 1737
tensor_r2_d3_sym_max (function), 1737
tensor_r2_d3_sym_property_count (function), 1737
tensor_r2_d3_sym_read_raw (subroutine), 1738
tensor_r2_d3_sym_reset (subroutine), 1738
tensor_r2_d3_sym_scalar_divide (function), 1738
tensor_r2_d3_sym_scalar_multiply (function), 1738
tensor_r2_d3_sym_scalar_multiply_switched (function), 1738
tensor_r2_d3_sym_serialize (subroutine), 1738
tensor_r2_d3_sym_set_to_identity (subroutine), 1738
tensor_r2_d3_sym_set_to_unity (subroutine), 1738
tensor_r2_d3_sym_subtract (function), 1738
tensor_r2_d3_sym_to_matrix (function), 1738
tensors (module), 1735
test_abundances (program), 1899
test_array_monotonicity (program), 1899
test_black_hole_fundamentals (program), 1899
test_comparison (program), 1900
test_coordinate_systems (program), 1900
test_hashes (program), 1901
test_hashes_cryptographic (program), 1901
test_inoue2014 (program), 1908
test_integration (program), 1901
test_integration_functions (module), 1902
test_interpolation (program), 1903
test_interpolation_2d (program), 1902
test_make_ranges (program), 1904
test_mass_distributions (program), 1904
test_math_special_functions (program), 1905
test_meshes (program), 1905
test_nfw96_concentration_dark_energy (program), 1896
test_node_task (subroutine), 1905
test_nodes (program), 1905
test_nodes_tasks (module), 1905
test_ode_solver (program), 1897
test_ode_solver_functions (module), 1897
test_perfect_hashes (program), 1901
test_prada2011_concentration (program), 1897
test_random (program), 1906
test_root_finding (program), 1907
test_root_finding_functions (module), 1907
test_search (program), 1907
test_sort (program), 1908

test_string_utilities (program), 1911
test_tables (program), 1911
test_tensors (program), 1912
test_vectors (program), 1912
test_zhao2009_dark_energy (program), 1898
test_zhao2009_flat (program), 1897
test_zhao2009_open (program), 1898
testfuncdouble0 (function), 1906
tests.abundances.F90 (file), 1899
tests.arrays.F90 (file), 1899
tests.black_hole_fundamentals.F90 (file), 1899
tests.bug745815.F90 (file), 1899
tests.comoving_distance.F90 (file), 1899
tests.comparisons.F90 (file), 1900
tests.cosmic_age.F90 (file), 1900
tests.geometry.coordinate_systems.F90 (file), 1900
tests.halo_mass_function.Tinker.F90 (file), 1900
tests.hashes.cryptographic.F90 (file), 1901
tests.hashes.F90 (file), 1901
tests.hashes.perfect.F90 (file), 1901
tests.integration.F90 (file), 1901
tests.integration.functions.F90 (file), 1902
tests.interpolation.2D.F90 (file), 1902
tests.interpolation.F90 (file), 1903
tests.IO.HDF5.F90 (file), 1896
tests.IO.XML.F90 (file), 1896
tests.kepler_orbits.F90 (file), 1903
tests.linear_growth.cosmological_constant.F90 (file), 1903
tests.linear_growth.dark_energy.F90 (file), 1903
tests.linear_growth.Eds.F90 (file), 1903
tests.linear_growth.open.F90 (file), 1904
tests.make_ranges.F90 (file), 1904
tests.mass_distributions.F90 (file), 1904
tests.math_special_functions.F90 (file), 1904
tests.meshes.F90 (file), 1905
tests.NFW96_concentration darken.F90 (file), 1896
tests.nodes.F90 (file), 1905
tests.nodes.task.F90 (file), 1905
tests.ODE_solver.F90 (file), 1896
tests.ODE_solver.functions.F90 (file), 1897
CODE INDEX

tests.power_spectrum.F90 (file), 1906
tests.Prada2011_concentration.F90 (file), 1897
tests.random.F90 (file), 1906
tests.regular_expressions.F90 (file), 1906
tests.root_finding.F90 (file), 1906
tests.root_finding.functions.F90 (file), 1907
tests.search.F90 (file), 1907
tests.sigma.F90 (file), 1907
tests.sort.F90 (file), 1908
tests.spectra.postprocess.Inoue2014.F90 (file), 1908
tests.sphericalcollapse.dark_energy.constantEoSminus0.6.F90 (file), 1908
tests.sphericalcollapse.dark_energy.constantEoSminus0.8.F90 (file), 1909
tests.sphericalcollapse.dark_energy.constantEoSminusHalf.F90 (file), 1909
tests.sphericalcollapse.dark_energy.constantEoSminusTwoThirds.F90 (file), 1909
tests.sphericalcollapse.dark_energy.EdS.F90 (file), 1908
tests.sphericalcollapse.dark_energy.lambda.F90 (file), 1910
tests.sphericalcollapse.dark_energy.open.F90 (file), 1910
tests.sphericalcollapse.flat.F90 (file), 1910
tests.sphericalcollapse.open.F90 (file), 1911
tests.string_utilities.F90 (file), 1911
tests.tables.F90 (file), 1911
tests.tree_branch_destroy.F90 (file), 1912
tests.vectors.F90 (file), 1912
tests.Zhao2009_algorithms.dark_energy.F90 (file), 1898
tests.Zhao2009_algorithms.EdS.F90 (file), 1897
tests.Zhao2009_algorithms.open.F90 (file), 1898
tests_bug745815 (program), 1899

tests.comoving_distance (program), 1900
tests.cosmic_age (program), 1900
tests_excursion_sets (program), 882
tests_halo_mass_function_tinker (program), 1901
tests_io_hdf5 (program), 1896
tests_io_xml (program), 1896
tests_kepler_orbits (program), 1903
tests_linear_growth.cosmological_constant (program), 1903
tests_linear_growth.dark_energy (program), 1904
tests_linear_growth.eds (program), 1903
tests_linear_growth.open (program), 1904
tests_power_spectrum (program), 1906
tests_regular_expressions (program), 1906
tests_sigma (program), 1908

tests_sphericalcollapse.dark_energy.eds (program), 1908
.tests_sphericalcollapse.dark_energy.lambda (program), 1910
tests_sphericalcollapse.dark_energy.oemga_half (program), 1909

tests_sphericalcollapse.dark_energy.oemga_two_thirds (program), 1910

tests_sphericalcollapse.dark_energy.oemga_zero_point_eight (program), 1909

tests_sphericalcollapse.dark_energy.oemga_zero_point_five (program), 1909

tests_sphericalcollapse.dark_energy.open (program), 1910
tests_sphericalCollapseFlat (program), 1911
tests_sphericalCollapseOpen (program), 1911
tests_treeBranchDestroy (program), 1912
 testvoidfunc (subroutine), 1906
thermodynamics.ideal_gases.F90 (file), 1912
thermodynamics.radiation.F90 (file), 1913
thermodynamics_radiation (module), 1913
tidal_heating_rate_gnedin (module), 1780
tidal_heating_rate_null (module), 1781
tidal_radius_heated_halo_solver (function), 1783
tidal_radius_root (function), 926
tidal_stripping_mass_loss_rate.disks.F90 (file), 1913
tidal_stripping_mass_loss_rate.disks.null.F90 (file), 1913
tidal_stripping_mass_loss_rate.disks.simple.F90 (file), 1914
tidal_stripping_mass_loss_rate.spheroids.F90 (file), 1915
tidal_stripping_mass_loss_rate.spheroids.null.F90 (file), 1915
tidal_stripping_mass_loss_rate.spheroids.simple.F90 (file), 1916
tidal_stripping_mass_loss_rate_disk (function), 1913
tidal_stripping_mass_loss_rate_disk_null (function), 1914
tidal_stripping_mass_loss_rate_disk_simple (function), 1914
tidal_stripping_mass_loss_rate_disks (module), 1913
tidal_stripping_mass_loss_rate_disks_null (module), 1913
tidal_stripping_mass_loss_rate_disks_null_init (subroutine), 1914
tidal_stripping_mass_loss_rate_disks_simple (module), 1914
tidal_stripping_mass_loss_rate_disks_simple_init (subroutine), 1914
tidal_stripping_mass_loss_rate_spheroid (function), 1915
tidal_stripping_mass_loss_rate_spheroid_null (function), 1915
tidal_stripping_mass_loss_rate_spheroid_simple (function), 1916
tidal_stripping_mass_loss_rate_spheroids (module), 1915
tidal_stripping_mass_loss_rate_spheroids_null (module), 1915
tidal_stripping_mass_loss_rate_spheroids_null_init (subroutine), 1915
tidal_stripping_mass_loss_rate_spheroids_simple (module), 1916
tidal_stripping_mass_loss_rate_spheroids_simple_init (subroutine), 1916
tidal_stripping_rate_null (module), 1783
tidal_stripping_rate_zentner2005 (module), 1782
time_ofCollapse (function), 1845
time_step_get (function), 1279
time_until_merging_subresolution (subroutine), 1250
timesteprange (type), 1143
transfer_function (function), 1882
transfer_function_bbks (module), 1880
transfer_function_bbks_initialize (subroutine), 1880
transfer_function_bbks_make (subroutine), 1880
transfer_function_bbks_state_retrieve (subroutine), 1880
transfer_function_bbks_state_store (subroutine), 1880
transfer_function_camb (module), 1881
transfer_function_camb_initialize (subroutine), 1881
transfer_function_camb_label (function), 1953
transfer_function_camb_make (subroutine), 1881
CODE INDEX

transfer_function_eisenstein_hu (module), 1881
transfer_function_eisenstein_hu_initialize (subroutine), 1881
transfer_function_eisenstein_hu_make (subroutine), 1882
transfer_function_eisenstein_hu_state_retrieve (subroutine), 1882
transfer_function_eisenstein_hu_state_store (subroutine), 1882
transfer_function_file_initialize (subroutine), 1884
transfer_function_file_read (subroutine), 1884
transfer_function_half_mode_mass (function), 1883
transfer_function_half_mode_mass_eisenstein_hu (function), 1882
transfer_function_half_mode_mass_null (function), 1880, 1881, 1884
transfer_function_initialize (subroutine), 1883
transfer_function_logarithmic_derivative (function), 1883
transfer_function_named_file_format_version (function), 1884
transfer_function_named_file_read (subroutine), 1884
transfer_function_null (module), 1884
transfer_function_null_initialize (subroutine), 1885
transfer_function_null_make (subroutine), 1885
transfer_function_state_retrieve (subroutine), 1883
transfer_function_tabulate_template (interface), 1883
transfer_function_tabulate_template (subroutine), 1883
transfer_functions (module), 1882
transfer_functions_file (module), 1883
tree_branch_mass (function), 1231
tree_branch_mass_template (function), 1231
tree_branch_mass_template (interface), 1231
tree_branching_initialize (subroutine), 1231
tree_branching_modifiers_initialize (subroutine), 1236
tree_branching_probability (function), 1231
tree_branching_probability_template (function), 1232
tree_branching_probability_template (interface), 1231
tree_maximum_step (function), 1232
tree_maximum_step_template (function), 1232
tree_maximum_step_template (interface), 1232
tree_node_component_builder (subroutine), 1665
tree_node_compute_derivatives (subroutine), 1276
tree_node_constructor (function), 1666
tree_node_copy_node_to (subroutine), 1666
tree_node_create_event (function), 1666
tree_node_create_initialize (subroutine), 1666
tree_node_evolve (subroutine), 1277
tree_node_evolve_error_analyzer (subroutine), 1277
tree_node_evolve_initialize (subroutine), 1278
tree_node_get_earliest_progenitor (function), 1666
tree_node_get_last_satellite (function), 1666
tree_node_index (function), 1666
tree_node_index_set (subroutine), 1666
tree_node_is_accurate (function), 1278
tree_node_is_on_main_branch (function), 1666
tree_node_is_primary_progenitor (function), 1666
tree_node_is_primary_progenitor_of_index (function), 1667
tree_node_is_primary_progenitor_of_node (function), 1667
tree_node_is_progenitor_of_index (function), 1667
tree_node_is_progenitor_of_node (function), 1667
tree_node_is_satellite (function), 1667
tree_node_merges_with_node (function), 1667
tree_node_move_components (subroutine), 1667
tree_node_ode_step_rates_initialize (subroutine), 1667
tree_node_ode_step_scales_initialize (subroutine), 1667
tree_node_odes (function), 1278
tree_node_odes_error_handler (subroutine), 1278
tree_node_promote (subroutine), 1278
tree_node_remove_from_host (subroutine), 1667
tree_node_remove_from_mergee (subroutine), 1668
tree_node_remove_paired_event (subroutine), 1668
tree_node_type (function), 1668
tree_node_unique_id (function), 1668
tree_node_unique_id_set (subroutine), 1668
tree_node_walk_branch (subroutine), 1668
tree_node_walk_branch_with_satellites (subroutine), 1668
tree_node_walk_tree (subroutine), 1669
tree_node_walk_tree_under_construction (subroutine), 1670
tree_node_walk_tree_with_satellites (subroutine), 1670
tree_subresolution_fraction (function), 1232
tree_subresolution_fraction_template (function), 1232
tree_subresolution_fraction_template (interface), 1232
treeevent (type), 1670
treemetadata (type), 1360
treenode (type), 1670
treenodedestroy (subroutine), 1670
treenodeinitialize (subroutine), 1670
treenodelist (type), 1670
triangular_shaped_cloud_integral (function), 1313
trigonometric_functions (module), 1229
trim (interface), 1224
trim (function), 1224

unit_tests (module), 1983
unit_tests_begin_group (subroutine), 1986
unit_tests_end_group (subroutine), 1986
unit_tests_finish (subroutine), 1986
unitsmetadata (type), 1360
universe (type), 1671
universecreateevent (function), 1671
universeevent (type), 1671
universeopttree (function), 1671
universepushtree (subroutine), 1671
universeremoveevent (subroutine), 1671
utility.arrays.F90 (file), 1930
utility.command_arguments.F90 (file), 1933
utility.date_and_time.F90 (file), 1934
utility.files.F90 (file), 1934
utility.hashes.cryptographic.F90 (file), 1938
utility.hashes.F90 (file), 1936
utility.hashes.perfect.F90 (file), 1938
utility.input_parameters.F90 (file), 1939
utility.IO.HDF5.F90 (file), 1916
utility.IO.XML.F90 (file), 1928
utility.kind_numbers.F90 (file), 1953
utility.memory_management.F90 (file), 1955
utility.memory_usage.cpp (file), 1975
utility.regular_expressions.F90 (file), 1975
utility.semaphores.F90 (file), 1975
utility.stateful_values.F90 (file), 1976
utility.string_handling.F90 (file), 1977
utility.unit_tests.F90 (file), 1983

validate_isolated_halos (subroutine), 1250
value_integer_scalar_ch (function), 1937
value_integer_scalar_i (function), 1937
value_integer_scalar_vs (function), 1938
values_agree (interface), 1294
values_agree_double (function), 1294
values_agree_real (function), 1294
values_differ (interface), 1294
values_differ_double (function), 1294
values_differ_real (function), 1295
values_integer_scalar (subroutine), 1938
var_str (interface), 1224
var_str_ (function), 1224
variance_integral (function), 1842
variance_integrand (function), 1843
variance_integrand_tophat (function), 1843
varying_string (type), 1224
vector_magnitude (function), 1230
vector_outer_product (function), 1230
vector_product (function), 1230
vector_self_outer_product (function), 1231
vectors (module), 1230
velocity_dispersion_integrand (function), 1085
verify (interface), 1224
verify_ch_vs (function), 1224
verify_vs_ch (function), 1224
verify_vs_vs (function), 1224
villalobos2013defaultconstructor (function), 1766, 1770
villalobos2013destructor (subroutine), 1766, 1770
villalobos2013genericconstructor (function), 1766, 1770
villalobos2013timeuntilmerging (function), 1766, 1770
virial_density_contrast (module), 1886
virial_orbital_parameters (function), 1774
virial_orbital_parameters_benson2005 (function), 1773
virial_orbital_parameters_benson2005_initialize (subroutine), 1773
virial_orbital_parameters_benson2005_snapshot (subroutine), 1773
virial_orbital_parameters_benson2005_state_retrieve (subroutine), 1773
virial_orbital_parameters_benson2005_state_store (subroutine), 1774
virial_orbital_parameters_fixed (function), 1776
virial_orbital_parameters_fixed_initialize (subroutine), 1776
virial_orbital_parameters_wetzel2010 (function), 1775
virial_orbital_parameters_wetzel2010_initialize (subroutine), 1775
virial_orbital_parameters_wetzel2010_snapshot (subroutine), 1775
virial_orbital_parameters_wetzel2010_state_retrieve (subroutine), 1775
virial_orbital_parameters_wetzel2010_state_store (subroutine), 1775
virial_orbits (module), 1774
virial_orbits_benson2005 (module), 1773
virial_orbits_fixed (module), 1775
virial_orbits_wetzel2010 (module), 1774
virialdensitycontrast (interface), 1889
virialdensitycontrastbryannorman1998 (interface), 1885, 1889
virialdensitycontrastclass (type), 1889
virialdensitycontrastconstructordefault (function), 1889
virialdensitycontrastconstructornamed (function), 1889
virialdensitycontrastdefinitioncalculationreset (subroutine), 1013, 1015
virialdensitycontrastdefinitionconstructor (function), 1013, 1015
virialdensitycontrastdefinitiondefaultconstructor (function), 1013, 1015
virialdensitycontrastdefinitiondestructor (subroutine), 1014, 1015
virialdensitycontrastdefinitiondynamicaltimescale (function), 1014, 1016
virialdensitycontrastdefinitionmeandensity (function), 1014, 1016
virialdensitycontrastdefinitionmeandensitygrowthrate (function), 1014, 1016
virialdensitycontrastdefinitionstaterestore (subroutine), 1014, 1016
virialdensitycontrastdefinitionstatestore (subroutine), 1014, 1016
virialdensitycontrastdefinitionvirialradius (function), 1014, 1016
virialdensitycontrastdefinitionvirialradiusgrowthrate (function), 1014, 1016
virialdensitycontrastdefinitionvirialtemperature (function), 1014, 1016
virialdensitycontrastdefinitionvirialvelocity (function), 1015, 1017
virialdensitycontrastdefinitionvirialvelocitygrowthrate (function), 1015, 1017
virialdensitycontrastdensitycontrasteofchangenu (function), 1890
virialdensitycontrastdensitycontrasteofchange (function), 1890
virialdensitycontrastdostateretrieve (subroutine), 1890
virialdensitycontrastdostatestore (subroutine), 1890
virialdensitycontrastfixed (interface), 1890, 1892
virialdensitycontrastfriendsoffriends (interface), 1890, 1893
virialdensitycontrastinitialize (subroutine), 1890
virialdensitycontrastisfinalizable (function), 1890
virialdensitycontrastsphericalcollapsematterde (interface), 1891, 1893
virialdensitycontrastsphericalcollapsematterlambda (interface), 1891, 1895
virialdensitycontraststaterestore (subroutine), 1891
virialdensitycontraststatestore (subroutine), 1891
virialfractionconstructor (function), 1172, 1173
virialfractiondefaultconstructor (function), 1172, 1174
virialfractiondestructor (subroutine), 1172, 1174
virialfractionradius (function), 1172, 1174
volume_integrand (function), 1819

wdmconcentration (function), 1057, 1061
wdmdefaultconstructor (function), 1057, 1062
wdmgenericconstructor (function), 1057, 1062
wetzelwhite2010defaultconstructor (function), 1766, 1770
wetzelwhite2010destructor (subroutine), 1766, 1770
wetzelwhite2010timeuntilmerging (function), 1766, 1770
wind_energy_integrand (function), 1821
write_parameter_xml (subroutine), 1953

xermsg (subroutine), 884
xi_integrand (function), 1245
xml_array_length (function), 1929
xml_array_read (interface), 1929
xml_array_read_one_column (subroutine), 1929
xml_array_read_static (interface), 1929
xml_array_read_static_one_column (subroutine), 1929
xml_array_read_two_column (subroutine), 1929
xml_extrapolation_element_decode (subroutine), 1929
xml_get_first_element_by_tag_name (function), 1929
xml_list_array_read_one_column (subroutine), 1930
xml_list_character_array_read_static_one_column (subroutine), 1930
xml_list_double_array_read_static_one_column (subroutine), 1930
xml_list_integer_array_read_static_one_column (subroutine), 1930
xml_path_exists (function), 1930
xray_absorption_ism_wilms2000 (program), 884
XRay_Absorption_ISM_Wilms2000.F90 (file), 884
xwrite (subroutine), 886

zhao2009concentration (function), 1057, 1062
zhao2009darkmatterprofiledefinition (function), 1057, 1062
zhao2009defaultconstructor (function), 1057, 1062
zhao2009densitycontrastdefinition (function), 1057, 1062

2198