A semi-analytic galaxy formation code.
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

5.1.7. Optional Outputs

5.2. Perl Module for Data Extraction

5.2.1. Derived Properties

5.3. Topics in Analysis of GALACTICUS Outputs

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

6. Input Data

6.1. Broadband Filters

6.2. Merger Trees

6.2.1. Processing of Merger Tree Files

6.2.2. Setting of Halo Properties

6.3. Requirements for GALACTICUS Input Parameters

7. Tutorials

7.1. Running GALACTICUS on N-body Merger Trees

7.1.1. Setting Input Parameters

7.1.2. Node Positions

7.1.3. Virial Orbits

7.1.4. Merging Times and Targets

7.1.5. Subhalo Masses

7.1.6. Node Spins

7.1.7. Node Scale Radii
11.7. Satellite Orbit ................................................. 169
  11.7.1. “Preset” Implementation ............................... 169
    Properties ..................................................... 169
    Initialization ............................................... 169
    Differential Evolution .................................... 169
    Event Evolution ........................................... 170
  11.7.2. “Very Simple” Implementation ......................... 170
    Properties ..................................................... 170
    Initialization ............................................... 170
    Differential Evolution .................................... 170
    Event Evolution ........................................... 170
  11.7.3. “Simple” Implementation ............................... 170
    Properties ..................................................... 170
    Initialization ............................................... 170
    Differential Evolution .................................... 171
    Event Evolution ........................................... 171
11.8. Dark Matter Halo Spin ....................................... 171
  11.8.1. “Random” Implementation .............................. 171
    Properties ..................................................... 171
    Initialization ............................................... 171
    Differential Evolution .................................... 171
    Event Evolution ........................................... 171
  11.8.2. “Preset” Implementation ............................... 172
    Properties ..................................................... 172
    Initialization ............................................... 172
    Differential Evolution .................................... 172
    Event Evolution ........................................... 172
11.9. Dark Matter Profile ......................................... 172
  11.9.1. “Scale” Implementation ............................... 172
    Properties ..................................................... 172
    Initialization ............................................... 172
    Differential Evolution .................................... 172
    Event Evolution ........................................... 173
  11.9.2. “Scale+Shape” Implementation ......................... 173
    Properties ..................................................... 173
    Initialization ............................................... 173
    Differential Evolution .................................... 173
    Event Evolution ........................................... 173
11.10. Merging Statistics ......................................... 173
  11.10.1. “Standard Implementation” ........................... 173
    Properties ..................................................... 173
    Initialization ............................................... 174
    Differential Evolution .................................... 174
    Event Evolution ........................................... 174
  11.10.2. “Recent” Implementation .............................. 174
    Properties ..................................................... 174
    Initialization ............................................... 174
    Differential Evolution .................................... 175
11.11 Formation Times ................................................................. 175
  11.11.1 “Cole2000” Implementation .......................................... 175
    Properties ........................................................................... 175
    Initialization ....................................................................... 175
    Differential Evolution .......................................................... 175
    Event Evolution .................................................................... 175
11.12 Indices ............................................................................... 175
  11.12.1 “Standard” Implementation ........................................... 176
    Properties ........................................................................... 176
    Initialization ....................................................................... 176
    Differential Evolution .......................................................... 176
    Event Evolution .................................................................... 176
11.13 Inter-Output Quantities ...................................................... 176
  11.13.1 “Standard” Implementation ........................................... 176
    Properties ........................................................................... 176
    Initialization ....................................................................... 176
    Differential Evolution .......................................................... 176
    Event Evolution .................................................................... 177

12. Physical Implementations ...................................................... 179
  12.1 Accretion of Gas into Halos .................................................. 179
    12.1.1 Simple Method ............................................................. 179
    12.1.2 Null Method .................................................................. 179
  12.2 Background Cosmology ....................................................... 179
    12.2.1 Matter + Lambda ......................................................... 180
    12.2.2 Matter + Dark Energy .................................................... 180
  12.3 Circumnuclear Accretion Disks ............................................. 180
    12.3.1 Shakura-Sunyaev Geometrically Thin, Radiatively Efficient Disks
    12.3.2 Advection Dominated, Geometrically Thick, Radiatively Inefficient Flows (ADAFs)
    12.3.3 “Switched” Disks ........................................................... 181
    12.3.4 Eddington-limited Disks .................................................. 181
  12.4 Dark Matter Structure Formation ......................................... 181
    12.4.1 Primordial Power Spectrum ......................................... 181
      (Running) Power Law Spectrum ......................................... 182
    12.4.2 Cosmological Mass Root Variance .................................. 182
      Filtered Power Spectrum .................................................... 182
    12.4.3 Power Spectrum Variance Window Function .................... 182
      Top-hat ............................................................................ 182
      Sharp in $k$-space ............................................................... 182
      Hybrid of Top-hat and Sharp in $k$-space ............................... 183
    12.4.4 Non-linear Matter Power Spectrum ................................ 183
      Linear ............................................................................. 183
      Peacock & Dodds (1996) ..................................................... 183
      CosmicEmu ...................................................................... 183
    12.4.5 Transfer Function .......................................................... 183
      Null ................................................................................. 183
      BBKS .............................................................................. 184
      Eisenstein & Hu ................................................................. 184
      CMBFAST ..................................................................... 184
File .......................................................... 184

12.4.6. Linear Growth Function .................................. 184
  Simple ....................................................... 184

12.4.7. Critical Overdensity ..................................... 184
  Spherical Collapse (Matter + Cosmological Constant) ............ 185
  Kitayama & Suto (1996) ....................................... 185

12.4.8. Critical Overdensity Mass Scaling ......................... 185
  Null .......................................................... 185
  Warm Dark Matter .......................................... 185

12.4.9. Virial Density Contrast ................................... 185
  Fixed ......................................................... 186
  Spherical Collapse (Matter + Cosmological Constant) .......... 186
  Kitayama & Suto (1996) ....................................... 186

12.4.10. Halo Bias ................................................ 186
  Press-Schechter ............................................. 186
  Sheth-Tormen ................................................ 186
  Tinker ......................................................... 186

12.4.11. Halo Mass Function .................................... 186
  Press-Schechter ............................................. 187
  Sheth-Tormen ................................................ 187
  Tinker ......................................................... 187

12.5. Cooling of Gas Inside Halos ................................ 187

12.5.1. Cooling Function ........................................ 187
  Atomic Collisional Ionization Equilibrium Using CLOUDY .......... 187
  Collisional Ionization Equilibrium From File ..................... 187
  CMB Compton Cooling ........................................ 188
  Molecular Hydrogen (Galli-Palla) ................................ 188

12.5.2. Cooling Rate ............................................ 188
  Cole et al. (2000) ........................................... 188
  Simple ....................................................... 189
  Simple Scaling .............................................. 189
  White & Frenk ............................................... 189

12.5.3. Cooling Rate Modifier ................................... 189
  Cut-Off ....................................................... 189

12.5.4. Cooling Radius ......................................... 189
  Simple ....................................................... 189
  Isothermal .................................................. 190

12.5.5. Cooling: Freefall Radius ................................. 190
  Dark Matter Halo ............................................ 190

12.5.6. Cooling: Infall Radius .................................. 190
  Cooling Radius ............................................... 190
  Cooling and Freefall Radii .................................. 190
  Dark Matter Halo ............................................ 190

12.5.7. Cooling Specific Angular Momentum........................ 190
  Constant Rotation ........................................... 191
  Mean ........................................................ 191

12.5.8. Cooling Time ............................................ 191
  Simple ....................................................... 191
12.5.9. Time Available for Cooling ........................................ 191
    Halo Formation ..................................................................... 191
    White & Frenk (1991) ......................................................... 192
12.5.10. Time Available for Freefall During Cooling ...................... 192
    Halo Formation ..................................................................... 192
12.6. Cosmology .................................................................. 192
    12.6.1. Matter + Cosmological Constant Universes .................... 192
12.7. Dark Matter Halos ...................................................... 192
    12.7.1. Mass Accretion History ........................................... 192
    Wechsler et al. (2002) ......................................................... 192
    Zhao et al. (2009) ............................................................... 193
    12.7.2. Density Profile ....................................................... 193
    Isothermal ......................................................................... 193
    NFW ................................................................................. 193
    Einasto .............................................................................. 193
    12.7.3. Density Profile Concentration .................................... 194
    Navarro, Frenk & White (1996) ........................................... 194
    Gao (2008) ......................................................................... 194
    Zhao (2009) ....................................................................... 194
    Muñoz-Cuartas (2011) ......................................................... 195
    Prada et al. (2011) .............................................................. 195
    12.7.4. Density Profile Shape .............................................. 195
    Gao (2008) ......................................................................... 195
    12.7.5. Mass Loss Rates ..................................................... 196
    Null .................................................................................. 196
    van den Bosch et al. (2005) ................................................ 196
    12.7.6. Mass Sampling Density Function ................................. 196
    Power Law ......................................................................... 196
    Halo Mass Function .......................................................... 196
    Stellar Mass Function ......................................................... 197
    12.7.7. Spin Parameter Distribution ...................................... 197
    Lognormal ......................................................................... 197
    Bett et al. (2007) ............................................................... 197
    Delta Function ................................................................... 197
12.8. Disk Stability/Bar Formation ........................................... 197
    12.8.1. Efstathiou, Lake & Negroponte ................................ 197
12.9. Excursion Set Barrier .................................................. 198
    12.9.1. Linear Barrier ........................................................ 198
    12.9.2. Quadratic Barrier ................................................... 198
    12.9.3. Critical Overdensity Barrier ..................................... 198
    12.10. Excursion Set Barrier First Crossing Distribution .......... 198
    12.10.1. Linear Barrier ........................................................ 198
    12.10.2. Farahi ................................................................. 199
    12.10.3. Zhang & Hui (2006) .................................................. 201
12.11. Excursion Set Barrier Remapping .................................. 203
    12.11.1. Null Remapping ................................................... 203
    12.11.2. Scale Remapping .................................................. 203
    12.11.3. Sheth-Mo-Tormen Remapping ................................. 203
12.12 Galactic Structure .................................................. 203
  12.12.1 Fixed ............................................................ 204
  12.12.2 Linear .......................................................... 204
  12.12.3 Simple .......................................................... 204
  12.12.4 Adiabatic ....................................................... 204
12.13 Galactic Structure Initial Radii .................................. 204
  12.13.1 Static ........................................................... 205
  12.13.2 Adiabatic ....................................................... 205
12.14 Galaxy Merging ...................................................... 205
  12.14.1 Mass Movements ................................................ 205
    Very Simple ......................................................... 206
    Simple ............................................................... 206
    Baugh et al. (2005) ................................................ 206
  12.14.2 Remnant Sizes ................................................ 206
    Null ................................................................. 206
    Cole et al. (2000) ............................................... 207
    Covington et al. (2008) ......................................... 207
  12.14.3 Progenitor Properties ....................................... 208
    Cole et al. (2000) ............................................... 208
    Standard ........................................................... 209
12.15 Hot Halo Density Profile ......................................... 209
  12.15.1 Cored Isothermal ............................................. 209
  12.15.2 Null ............................................................ 210
12.16 Hot Halo Density Profile: Cored Isothermal Core Radius .... 210
  12.16.1 Growing Core ................................................ 210
  12.16.2 Virial Radius Fraction ....................................... 210
12.17 Hot Halo Ram Pressure Force .................................... 210
  12.17.1 Null ............................................................ 210
  12.17.2 Font et al. (2008) .......................................... 210
12.18 Hot Halo Ram Pressure Stripping Radius ......................... 210
  12.18.1 Virial Radius ................................................ 211
  12.18.2 Font et al. (2008) .......................................... 211
12.19 Hot Halo Temperature Profile .................................. 211
  12.19.1 Virial Temperature ........................................... 211
12.20 Initial Mass Functions ........................................... 211
  12.20.1 Initial Mass Function Selection .............................. 211
    Fixed ............................................................... 211
    Disk and Spheroid ............................................... 211
  12.20.2 Initial Mass Functions ....................................... 212
    Baugh et al. (2005) Top-Heavy ................................... 212
    Chabrier .......................................................... 212
    Kennicutt ........................................................ 212
    Kroupa ............................................................. 212
    Miller-Scalo ..................................................... 212
    Piecewise Power-law ............................................ 213
    Salpeter .......................................................... 213
    Scalo .............................................................. 213
12.21 Intergalactic Medium State ...................................... 213
  12.21.1 RecFast ........................................................ 213
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.21.2 File</td>
<td>213</td>
</tr>
<tr>
<td>12.22 Chemical State</td>
<td>213</td>
</tr>
<tr>
<td>12.22.1 Atomic Collisional Ionization Equilibrium Using Cloudy</td>
<td>214</td>
</tr>
<tr>
<td>12.22.2 Collisional Ionization Equilibrium From File</td>
<td>214</td>
</tr>
<tr>
<td>12.23 Merger Tree Construction</td>
<td>214</td>
</tr>
<tr>
<td>12.23.1 Read From File</td>
<td>214</td>
</tr>
<tr>
<td>12.23.2 Build</td>
<td>214</td>
</tr>
<tr>
<td>12.23.3 Fully Specified</td>
<td>215</td>
</tr>
<tr>
<td>12.23.4 Smooth Accretion</td>
<td>216</td>
</tr>
<tr>
<td>12.23.5 State Restore</td>
<td>216</td>
</tr>
<tr>
<td>12.24 Merger Tree Branching</td>
<td>217</td>
</tr>
<tr>
<td>12.24.1 Modified Press-Schechter</td>
<td>217</td>
</tr>
<tr>
<td>12.24.2 Generalized Press-Schechter</td>
<td>217</td>
</tr>
<tr>
<td>12.25 Merger Tree Branching Modifier</td>
<td>218</td>
</tr>
<tr>
<td>12.25.1 Null</td>
<td>218</td>
</tr>
<tr>
<td>12.25.2 Parkinson, Cole &amp; Helly (2008)</td>
<td>218</td>
</tr>
<tr>
<td>12.26 Merger Tree Building</td>
<td>218</td>
</tr>
<tr>
<td>12.27 Merger Tree Pre-evolution Processing</td>
<td>219</td>
</tr>
<tr>
<td>12.27.1 Enforce Monotonic Mass Growth</td>
<td>219</td>
</tr>
<tr>
<td>12.27.2 Interpolate Tree to Time Grid</td>
<td>219</td>
</tr>
<tr>
<td>12.27.3 Mass Accretion History Output</td>
<td>220</td>
</tr>
<tr>
<td>12.27.4 Tree Pruning By Mass</td>
<td>220</td>
</tr>
<tr>
<td>12.27.5 Tree Pruning By Hierarchy</td>
<td>220</td>
</tr>
<tr>
<td>12.28 Chemical Reaction Rates</td>
<td>220</td>
</tr>
<tr>
<td>12.28.1 Null</td>
<td>220</td>
</tr>
<tr>
<td>12.28.2 Hydrogen Network</td>
<td>220</td>
</tr>
<tr>
<td>12.29 Ram Pressure Induced Mass Loss Rates in Disks</td>
<td>220</td>
</tr>
<tr>
<td>12.29.1 Simple</td>
<td>221</td>
</tr>
<tr>
<td>12.29.2 Null</td>
<td>221</td>
</tr>
<tr>
<td>12.30 Star Formation Rate Surface Densities</td>
<td>221</td>
</tr>
<tr>
<td>12.30.1 Kennicutt-Schmidt</td>
<td>221</td>
</tr>
<tr>
<td>12.30.2 Extended Schmidt</td>
<td>222</td>
</tr>
<tr>
<td>12.30.3 Blitz-Rosolowsky</td>
<td>222</td>
</tr>
<tr>
<td>12.30.4 Krumholz-McKee-Tumlinson</td>
<td>222</td>
</tr>
<tr>
<td>12.31 Star Formation Timescales</td>
<td>223</td>
</tr>
<tr>
<td>12.31.1 Fixed</td>
<td>223</td>
</tr>
<tr>
<td>12.31.2 Halo Scaling</td>
<td>223</td>
</tr>
<tr>
<td>12.31.3 Dynamical Time</td>
<td>223</td>
</tr>
<tr>
<td>12.31.4 Integrated Surface Density</td>
<td>224</td>
</tr>
<tr>
<td>12.31.5 Baugh et al. (2005)</td>
<td>224</td>
</tr>
<tr>
<td>12.32 Stellar Population Properties</td>
<td>224</td>
</tr>
<tr>
<td>12.32.1 Instantaneous</td>
<td>224</td>
</tr>
<tr>
<td>12.32.2 Noninstantaneous</td>
<td>224</td>
</tr>
<tr>
<td>12.33 Stellar Population Spectra</td>
<td>225</td>
</tr>
<tr>
<td>12.33.1 Conroy, White &amp; Gunn (2009)</td>
<td>225</td>
</tr>
<tr>
<td>12.33.2 File</td>
<td>225</td>
</tr>
<tr>
<td>12.34 Stellar Population Spectra Postprocessing</td>
<td>225</td>
</tr>
<tr>
<td>12.34.1 Meiksin (2006) IGM Attenuation</td>
<td>225</td>
</tr>
</tbody>
</table>
12.3.4.2. Madau (1995) IGM Attenuation .................................................. 225
12.3.4.3. Lyman-continuum Suppression .................................................. 226
12.3.4.4. Recent Star Formation ............................................................. 226
12.3.4.5. Identity ................................................................. 226
12.3.5. Stellar Astrophysics ............................................................. 226
12.3.5.1. Basics ................................................................. 226
12.3.5.2. Stellar Winds ............................................................. 227
Leitherer et al. (1992) ............................................................. 227
12.3.5.3. Stellar Tracks ............................................................. 227
File ................................................................. 227
12.3.5.4. Supernovae Type Ia ........................................................... 228
Nagashima et al. (2005) Prescription ............................................. 228
12.3.5.5. Population III Supernovae ...................................................... 228
Heger & Woosley (2002) ............................................................. 228
12.3.5.6. Stellar Feedback ............................................................. 228
Standard ................................................................. 228
12.3.6. Substructure and Merging ......................................................... 228
12.3.6.1. Merging Timescales .......................................................... 228
Dynamical Friction: Lacey & Cole ................................................. 229
Dynamical Friction: Lacey & Cole + Tormen .................................. 229
Dynamical Friction: Jiang (2008) .................................................... 229
Dynamical Friction: Wetzel & White (2010) .................................. 229
12.3.6.2. Virial Orbits ................................................................. 229
Benson (2005) ................................................................. 229
Fixed ................................................................. 229
Wetzel (2010) ................................................................. 230
12.3.6.3. Node Merging ................................................................. 230
Single Level Hierarchy ............................................................. 230
12.3.7. Supernovae Feedback Models .................................................... 230
12.3.7.1. Fixed ................................................................. 230
12.3.7.2. Power Law ................................................................. 230
12.3.7.3. Creasey et al. (2012) ....................................................... 231
12.3.8. Supernovae Expulsive Feedback Models ..................................... 231
12.3.8.1. Null ................................................................. 231
12.3.8.2. Superwind ................................................................. 231
12.3.9. Supermassive Black Hole Binaries: Initial Separation ................. 231
12.3.9.1. Spheroid Radius Fraction .................................................. 231
12.3.9.3. Tidal Radius ............................................................. 232
12.3.10. Supermassive Black Hole Binaries: Separation Growth Rate .... 232
12.3.10.1. Null ................................................................. 232
12.3.10.2. Standard ................................................................. 232
12.3.11. Supermassive Black Holes Binaries: Recoil Velocity ................. 233
12.3.11.1. Null ................................................................. 233
12.3.11.2. Campanelli et al. (2007) ................................................... 233
12.3.12. Supermassive Black Hole Binaries: Mergers ......................... 234
12.3.12.1. Rezzolla et al. (2008) ....................................................... 234
13. Additional Output Quantities
   13.1. Black Hole Accretion ................................................................. 235
   13.2. Cooling Data ............................................................................. 235
   13.3. Density Contrast Data ................................................................. 235
   13.4. Descendent Node Index ............................................................... 235
   13.5. Half-Light Radii Data ................................................................. 236
   13.6. Halo Model Quantities ................................................................. 236
   13.7. Lightcone Coordinates ............................................................... 236
   13.8. Main Branch Evolution ............................................................... 239
   13.9. Main Branch Status ................................................................. 239
   13.10. Mass Profile Data ................................................................. 239
   13.11. Merger Tree Links and Node Isolation ......................................... 239
   13.12. Merger Tree Data for Rendering ............................................... 240
   13.13. Merger Tree Structure ............................................................. 240
   13.15. Rotation Curve ......................................................................... 240
   13.16. Satellite Orbital Pericenter ........................................................ 242
   13.17. Star Formation Rates ............................................................... 242
   13.18. Velocity Dispersion ................................................................. 242
   13.19. Virial Quantities ...................................................................... 243

IV. Development .................................................................................... 245
14. Developing Galacticus ................................................................. 249
   14.1. Getting Started ........................................................................... 249
   14.1.1. Using BitBucket ...................................................................... 249
   14.2. Making Simple Changes .............................................................. 249
   14.2.1. Contributing Your Changes Back To Galacticus ....................... 250
         Via E-mail ...................................................................................... 250
         Using BitBucket ............................................................................ 250
   14.3. Making Bigger Changes .............................................................. 250
   14.4. Releases ..................................................................................... 251
   14.4.1. Bug Fixes In Releases .............................................................. 251

15. Coding Galacticus ............................................................................ 253
   15.1. Numerical Tools ........................................................................ 253
   15.1.1. Finding Roots of Equations ..................................................... 253
   15.2. Computation Dependencies and Data Files ................................. 255
   15.3. Optimization ............................................................................. 256
   15.3.1. Unique IDs and Stored Properties ....................................... 256
   15.4. Mixed Language Coding ............................................................ 256
   15.4.1. Component Property Methods .............................................. 256
   15.4.2. Using Functions in C++ ......................................................... 257
   15.4.3. Adding New Implementations in C++ .................................. 257
   15.5. Objects ....................................................................................... 257
   15.5.1. Enumerations ....................................................................... 257
         adjustElements ............................................................................ 258
         componentType ........................................................................... 258
         coordinateSystem ...................................................................... 258
16. Adding New Methods

16.1. Code Directives ................................................. 449
16.2. Identifying Components and Mass Types ....................... 449
16.3. Components .................................................... 450
   16.3.1. Component Structure ..................................... 450
   16.3.2. Extending Components ................................... 450
   16.3.3. Implementing a New Component
            Component Definition ..................................... 451
            Component Initialization ................................ 455
            Component Access, Creation and Destruction .......... 455
            Component Methods ....................................... 456
Halo Profiles ................................................................. 518
Halo Virial Density Contrast ........................................... 520
Initial Mass Function Functions ......................................... 521
Initial Mass Function Selection ......................................... 524
Intergalactic Medium State ............................................. 525
Linear Growth Function ................................................ 527
Merger Tree Branching .................................................. 528
Merger Tree Branching Modifiers ....................................... 531
Merger Tree Building .................................................... 531
Merger Tree Construction ................................................ 532
Non-linear Matter Power Spectrum ...................................... 534
Chemical Reaction Rates ............................................... 535
Population III Supernovae .............................................. 536
Power Spectrum Variance Window Function ......................... 537
Primordial Power Spectrum ........................................... 538
Radiation Components .................................................. 539
Radiation Components: Intergalactic Background ....................... 541
Ram Pressure Mass Loss Rates in Disks ............................... 543
Satellite Merging Mass Movements ..................................... 544
Satellite Merging Remnant Sizes ....................................... 545
Satellite Merging Remnants: Progenitor Properties ................... 546
Satellite Merging Timescales ........................................... 548
Satellite Virial Orbits .................................................... 549
Star Formation Feedback in Disks/Spheroids .......................... 550
(Expulsive) Star Formation Feedback in Disks/Spheroids ............... 551
Star Formation Rate Surface Density in Disks ......................... 552
Star Formation Timescale in Disks/Spheroids ......................... 553
Stellar Astrophysics ...................................................... 554
Stellar Population Properties .......................................... 555
Stellar Population Spectra ............................................. 558
Stellar Population Spectra Postprocessing .............................. 560
Star Feedback .................................................................. 561
Stellar Tracks ................................................................ 562
Stellar Winds ................................................................ 563
Supernovae Type Ia ....................................................... 564
Tree Timing .................................................................. 565
Transfer Function .......................................................... 566
16.4.2. Events ................................................................ 569
Node Merger Events ........................................................ 569
Node Promotion Events .................................................... 570
16.4.3. Tasks ................................................................ 570
Calculation Reset Tasks .................................................... 570
Cooling Rate Modifiers ..................................................... 571
Decode Property Identifier Tasks ......................................... 571
Evolution Timestep Tasks .................................................. 572
Galactic Component Density ............................................. 573
Galactic Component Enclosed Mass ....................................... 574
Galactic Component Rotation Curve ...................................... 574
Galactic Component Rotation Curve Gradient ......................... 575
## VI. Appendices

### A. Merger Tree File Format

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1. Basic File Format</td>
<td>1372</td>
</tr>
<tr>
<td>A.1.1. Flexibility and Extensibility</td>
<td>1372</td>
</tr>
<tr>
<td>A.1.2. A Note on Scalar Attributes</td>
<td>1372</td>
</tr>
<tr>
<td>A.1.3. Example File Structure</td>
<td>1372</td>
</tr>
<tr>
<td>A.2. Cosmology Group</td>
<td>1373</td>
</tr>
<tr>
<td>A.2.1. Standard Attributes</td>
<td>1374</td>
</tr>
<tr>
<td>A.3. Group Finder Group</td>
<td>1374</td>
</tr>
<tr>
<td>A.3.1. Standard Attributes</td>
<td>1375</td>
</tr>
<tr>
<td>A.4. Simulation Group</td>
<td>1375</td>
</tr>
<tr>
<td>A.4.1. Standard Attributes</td>
<td>1377</td>
</tr>
<tr>
<td>GADGET-specific Standard Attributes</td>
<td>1377</td>
</tr>
<tr>
<td>A.5. Units Group</td>
<td>1378</td>
</tr>
<tr>
<td>A.6. Halo Trees Group</td>
<td>1380</td>
</tr>
<tr>
<td>A.6.1. Standard Attributes</td>
<td>1382</td>
</tr>
<tr>
<td>A.6.2. Standard Datasets</td>
<td>1383</td>
</tr>
<tr>
<td>A.7. Tree Index Group</td>
<td>1384</td>
</tr>
<tr>
<td>A.7.1. Standard Datasets</td>
<td>1384</td>
</tr>
<tr>
<td>A.8. Merger Trees Group</td>
<td>1385</td>
</tr>
<tr>
<td>A.9. Particles Group</td>
<td>1385</td>
</tr>
<tr>
<td>A.9.1. Standard Datasets</td>
<td>1386</td>
</tr>
<tr>
<td>A.10. Merger Tree Builder</td>
<td>1387</td>
</tr>
<tr>
<td>A.10.1. File Formats</td>
<td>1390</td>
</tr>
<tr>
<td>A.10.2. Exporting Trees from GALACTICUS</td>
<td>1391</td>
</tr>
</tbody>
</table>

### B. Plotting Support

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1. Plotting with Gnuplot</td>
<td>1393</td>
</tr>
<tr>
<td>B.2. Merger Tree Diagrams with DOT</td>
<td>1395</td>
</tr>
</tbody>
</table>

### Glossary

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1409</td>
</tr>
</tbody>
</table>

### Acronyms

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1411</td>
</tr>
</tbody>
</table>
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 §16.1) 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” components (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. 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:

```bash
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\(^1\) 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:

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

\(^1\)Early versions of HDF5 may work, but versions prior to v1.8.5 have known memory leak problems.
2.3. Required Libraries and Tools

**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
```

Alternatively, it can be installed from CPAN using:

```
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 -MCPAN -e 'install List::Uniq'
```
2. Installation

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:

```bash
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:

```bash
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

```bash
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

```bash
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:

```bash
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\(^2\). It is used by analysis scripts to extract data from GALACTICUS output files. Follow the following instructions (as root) to install this module:

```bash
wget http://users.obs.carnegiescience.edu/abenson/galacticus/tools/PDL-IO-HDF5-0.6.tar.gz
tar xvfz PDL-IO-HDF5-0.6.tar.gz
cd PDL-IO-HDF5-0.6
perl Makefile.PL
make
make test
make install
```

\(^2\)This is a modified version of the original PDL::IO::HDF5 module. It supports creation and automatic de-referencing of HDF5 region references and also supports reading of numerical attributes and 1-D attribute arrays.
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'
```

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
```

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'
```

item [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. Installation

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'
```

**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-TermReadKey
```

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'
```

---

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 LaTeX::Encode.pm file on your system edit it to comment out (or remove) the “use strict;” line.
2.3. Required Libraries and Tools

```plaintext
yum install perl-DateTime
Alternatively, it can be installed from CPAN using:
perl -MCPAN -e 'install Date::Format'

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'

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:

```
maker 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.

- **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
```
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"
```
2. Installation

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

! Install zlib:

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

! Install HDF5:

wget http://www.hdfgroup.org/ftp/HDF5/current/src/hdf5-1.8.7.tar.bz2
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:

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
2.5. Installing Without Root Access

! Install Mercurial:

```
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:

```
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
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'
```
2. Installation

perl -MCORE -e 'install IO::Compress::Bzip2'
perl -MCORE -e 'install Date::Format'
perl -MCORE -e 'install Net::SMTP::SSL'
perl -MCORE -e 'install Net::DBus'
perl -MCORE -e 'install PDL'
perl -MCORE -e 'install PDL::LinearAlgebra'
perl -MCORE -e 'install PDL::MatrixOps'
perl -MCORE -e 'install PDL::NiceSlice'

! Install PDL::IO::HDF5:

wget http://users.obs.carnegiescience.edu/abenson/galacticus/tools/PDL-IO-HDF5-0.6.tar.gz

tar xvfz PDL-IO-HDF5-0.6.tar.gz

cd PDL-IO-HDF5-0.6
perl Makefile.PL
make
make test
make install

! 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_v0.9.2.tar.bz2"

tar xvfj galacticus_v0.9.2.tar.bz2

cd Galacticus/v0.9.2

! 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
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

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

and do the usual

```bash
$ 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

```bash
$ 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

```bash
$ sudo perl -MCPAN -e 'install PDF::Create'
```

and then

```bash
$ 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 login 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

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. 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:

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. The generated parameters are printed to standard output as Galacticus-compatible XML.

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.

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

3.3.1. 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:

```
<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:

```
<parameter>
  <name>stateRetrieveFileRoot</name>
  <value>galacticusState</value>
</parameter>
<parameter>
  <name>mergerTreeBuildTreesBeginAtTree</name>
  <value>N</value>
</parameter>
```

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.

3.3.2. 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/Run_Galacticus.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 or on a Condor cluster. To run the script simply enter:

```
./scripts/aux/Run_Galacticus.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
3. Running Galacticus

has finished. If multiple instances are to be launched on multiple machines a command line option to `Run_Galacticus.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 `Run_Galacticus.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>
    <threadCount>maximum</threadCount>
    <compressModels>no</compressModels>
    <condor>
        <useCondor>true</useCondor>
        <galacticusDirectory>/home/condor/Galacticus/v0.9.1</galacticusDirectory>
        <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>
        <wholeMachine>true</wholeMachine>
    </condor>
    <pbs>
        <usePBS>true</usePBS>
        <scratchPath>/scratch/me</scratchPath>
        <wallTime>48:00:00</wallTime>
        <memory>3gb</memory>
        <ompThreads>8</ompThreads>
        <queue>standard</queue>
        <maxJobsInQueue>10</maxJobsInQueue>
        <mpiRun>/opt/openmpi/bin/mpirun</mpiRun>
        <environment>LD_LIBRARY_PATH=/home/me/software/Galacticus/Tools/lib64:$LD_LIBRARY_PATH</environment>
    </pbs>
    <parameters>
        <parameter>
            <name>stabilityThresholdStellar</name>
            <value>1.1</value>
        </parameter>
        <parameter>
            <name>stabilityThresholdGaseous</name>
            <value>1.1</value>
        </parameter>
        <parameter>
            <name>imfSalpeterYieldInstantaneous</name>
            <value>0.02</value>
        </parameter>
    </parameters>
</parameterGrid>
```
3.3. Running Galacticus

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 [starFormationKennicuttSchmidtTruncate]=true and false only when [starFormationTimescaleDisksMethod]=Kennicutt-Schmidt and not when [starFormationTimescaleDisksMethod]=dynamical time.

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 §3.3.3);

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.

threadCount Specifies the number of threads that should be launched (each running a separate GALACTICUS model) when running on the local machine. If set to “maximum” then the number of threads will be set to the available number of cores on the local machine. If not present, a single thread is used.

compressModels If “no” then models are not compressed after being run. Otherwise, the contents of the model output directory will be compressed using bzip2.
3. Running Galacticus

condor This section, if present, specifies if and how jobs should be submitted to a Condor cluster. The following options are available:

useCondor If set to “true” then jobs will be submitted to a CONDOR cluster, otherwise they will be run on the local machine;

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;

evironment 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.

pbs This section, if present, specifies if and how jobs should be submitted to a PBS batch queue system. The following options are available:

usePBS If set to “true” then jobs will be submitted to a PBS batch queue system, otherwise they will be run on the local machine;

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 can be used).\(^2\)

\(^2\)As described here for example.
are spawned). In such cases, setting this option to “no” should permit multiple CPUs to be utilized.

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;

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 §3.3.1) for the run can also be found in the model directory.

### 3.3.3. 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 §3.3.2). 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:

```xml
<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>
```

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:
3. Running Galacticus

```xml
<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.

3.3.4. 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:

```xml
<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 “.hdf5”.

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.
3.3.5. Limiting the Load Average

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

3.4. Additional Codes

The \text{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 \text{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:

\text{make Excursion\_Sets.exe}
\text{Excursion\_Sets.exe <parameterFile> <outputFile>}

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

- \text{haloMass}
- \text{wavenumber}
- \text{powerSpectrum}
- \text{variance}
- \text{barrier}

These datasets contain the following information:

- \text{haloMass} Halo mass \([M_{\odot}]\);
- \text{wavenumber} Wavenumber corresponding to this halo mass \([\text{Mpc}^{-1}]\);
- \text{powerSpectrum} Power spectrum at this wavenumber \([\text{Mpc}^{3}]\);
- \text{variance} The variance, \(S(M) = \sigma^2(M)\), at this halo mass;
- \text{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 $[\text{Gyr}^{-1}]$ for all pairs of halo mass;

haloMassFunction The halo mass function $[\text{M}_\odot^{-1} \text{Mpc}^{-3}]$.

### 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:

```bash
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 $\text{M}_\odot$) at which to tabulate;
- `[haloMassFunctionsMassMaximum]` The highest mass halo (in units of $\text{M}_\odot$) 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:

```bash
+- Outputs 
  |  
  |  +-> outputCharacteristicMass [dataset] 
  |  
  |  +-> outputCriticalOverdensities [dataset] 
  |  
  |  +-> outputExpansionFactor [dataset] 
  |  
  |  +-> outputGrowthFactors [dataset] 
  |  
  |  +-> outputRedshift [dataset] 
  |  
  |  +-> outputTime [dataset] 
  |  
  |  +-> outputVirialDensityContrast [dataset] 
  
+- Parameters 
  |  
  |  +-> parameter1 [attribute] 
  |  
  |  +-> parameterN [attribute]
```
### 3.4. Additional Codes

<table>
<thead>
<tr>
<th>haloMassFunctions</th>
</tr>
</thead>
<tbody>
<tr>
<td>++-&gt; haloBias [dataset]</td>
</tr>
<tr>
<td>++-&gt; haloMass [dataset]</td>
</tr>
<tr>
<td>++-&gt; haloMassFractionCumulative [dataset]</td>
</tr>
<tr>
<td>++-&gt; haloMassFunctionCumulative [dataset]</td>
</tr>
<tr>
<td>++-&gt; haloMassFunctionLnM [dataset]</td>
</tr>
<tr>
<td>++-&gt; haloMassFunctionM [dataset]</td>
</tr>
<tr>
<td>++-&gt; haloNu [dataset]</td>
</tr>
<tr>
<td>++-&gt; haloSigma [dataset]</td>
</tr>
<tr>
<td>++-&gt; haloVirialRadius [dataset]</td>
</tr>
<tr>
<td>++-&gt; haloVirialTemperature [dataset]</td>
</tr>
<tr>
<td>++-&gt; haloVirialVelocity [dataset]</td>
</tr>
</tbody>
</table>

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 $M_\odot$), $M_*$, 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;
- **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 $\text{Mpc}^{-3}$);
3. Running Galacticus

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]
  |   |
  |   +-> wavenumber [dataset]
  |
+-> Parameters
  |
  +-> parameter1 [attribute]
  |
  +-> parameterN [attribute]
```

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 $\sigma(M)$: $\alpha = d \ln \sigma / d \ln M$;
mass The mass scale, $M$, corresponding to the given wavenumber, $k$, defined such that $M = 4\pi\Omega_M\rho_{\text{crit}}/3k^3$ (in units of $M_{\odot}$);
powerSpectrum The linear theory power spectrum at $z = 0$: $P(k)$ in units of Mpc$^3$;

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

wavenumber The wavenumber in units of 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: $H_0$
Attached to: module:Cosmological_Parameters
File: 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: $\Omega_{DE}$
Attached to: module:Cosmological_Parameters
File: 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: $\Omega_{Matter}$
Attached to: module:Cosmological_Parameters
File: 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: $\Omega_b$
Attached to: module:Cosmological_Parameters
File: 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: $T_{CMB}$
Attached to: module:Cosmological_Parameters
File: cosmology.parameters.F90
Default value: 2.72548 [Fixsen, 2009]
Description: The present day temperature of the cosmic microwave background (CMB) in units of Kelvin.

Name: accretionDiskJetPowerEddington
Attached to: module:Accretion_Disks_Eddington_limited
File: 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.
4. Input Parameters

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

Name: accretionDiskSwitchedScaleAdafRadiativeEfficiency
Attached to: module:Accretion_Disks_Switched
File: 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: accretionDisksMethod
Attached to: module:Accretion_Disks
File: accretion_disks.F90
Default value: switched
Description: Selects which accretion disk method should be used.

Name: accretionHalosMethod
Attached to: module:Accretion_Halos
File: accretion.halo.F90
Default value: simple
Description: Selects which method should be used for accretion onto halos.

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

Name: accretionHistoryWechslerFormationRedshiftCompute
Attached to: 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: accretionHistoryWechslerFormationRedshift
Attached to: 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: accretionRateThinDiskMaximum
Attached to: module:Accretion_Disks_Switched
File: 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:** accretionRateThinDiskMinimum  
**Attached to:** module:Accretion_Disks_Switched  
**File:** accretion_disks.switched.F90  
**Default value:** 0.01  
**Description:** The accretion rate (in Eddington units) below which a switched accretion disk becomes an ADAF.

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

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

**Name:** adafRadiativeEfficiencyType  
**Attached to:** module:Accretion_Disks_ADAF  
**File:** accretion_disks.ADAF.F90  
**Default value:** thinDisk
4. Input Parameters

**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:** adafRadiativeEfficiency  
**Attached to:** module:Accretion_Disks_ADAF  
**File:** 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:** adafViscosityFixedAlpha  
**Attached to:** module:Accretion_Disks_ADAF  
**File:** 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:** adafViscosityOption  
**Attached to:** module:Accretion_Disks_ADAF  
**File:** 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:** adiabaticContractionGnedinA  
**Attached to:** module:Galactic_Structure_Initial_Radii_Adiabatic  
**File:** 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:** adiabaticContractionGnedinOmega  
**Attached to:** module:Galactic_Structure_Initial_Radii_Adiabatic  
**File:** 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:** adiabaticContractionIncludeBaryonGravity  
**Attached to:** module:Galactic_Structure_Radii_Adiabatic  
**File:** 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:** adiabaticContractionSolutionTolerance  
**Attached to:** module:Galactic_Structure_Radii_Adiabatic  
**File:** galactic_structure.radius_solver.adiabatic.F90  
**Default value:** HASH(0x19195680)  
**Description:** Maximum allowed mean fractional error in the radii of all components when seeking equi-
librium solutions for galactic structure.

Name: adiabaticContractionUseFormationHalo
Attached to: module:Galactic_Structure_Radii_Adiabatic
File: galactic_structure.radius_solver.adiabatic.F90
Default value: false
Description: Specifies whether or not the “formation halo” should be used when solving for the radii of galaxies.

Name: allTreesExistAtFinalTime
Attached to: module:Merger_Trees_Evolve
File: 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: atomicCollisionalIonizationMethod
Attached to: module:Atomic_Rates_Ionization_Collisional
File: atomic.rates.ionization.collisional.F90
Default value: Verner
Description: The name of the method to be used for computing atomic collisional ionization rates.

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

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

Name: blackHoleAccretesFromHotHalo
Attached to: module:Node_Component_Black_Hole_Simple
File: 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: blackHoleBinariesComputeVelocityDispersion
Attached to: module:Black_Hole_Binary_Separations_Standard
4. Input Parameters

File: `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: `blackHoleBinaryInitialRadiiMethod`
Attached to: `module:Black_Hole_Binary_Initial_Radii`
File: `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: `blackHoleBinaryMergersMethod`
Attached to: `module:Black_Hole_Binary_Mergers`
File: `black_holes.binary_mergers.F90`
Default value: `Rezzolla2008`
Description: The name of the method to be used for computing the effects of black hole binary mergers.

Name: `blackHoleBinaryRecoilVelocityMethod`
Attached to: `module:Black_Hole_Binary_Recoil_Velocities`
File: `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: `blackHoleBinarySeparationGrowthRateMethod`
Attached to: `module:Black_Hole_Binary_Separations`
File: `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: `blackHoleHeatingEfficiency`
Attached to: `module:Node_Component_Black_Hole_Simple`
File: `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: `blackHoleHeatsHotHalo`
Attached to: `module:Node_Component_Black_Hole_Simple`
File: `objects.nodes.components.black_hole.simple.F90`
Default value: `true`
Description: Specifies whether or not the black hole should heat the hot halo.

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

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

Name: blackHoleToSpheroidStellarGrowthRatio
Attached to: module:Node_Component_Black_Hole_Simple
File: 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: blackHoleWindEfficiencyScalesWithRadiativeEfficiency
Attached to: module:Node_Component_Black_Hole_Standard
File: 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: blackHoleWindEfficiency
Attached to: module:Node_Component_Black_Hole_Simple
File: objects.nodes.components.black_hole.simple.F90
Default value: $2.2157 \times 10^{-3}$
Description: The efficiency of the black hole accretion-driven wind.

Name: bondiHoyleAccretionEnhancementHotHalo
Attached to: module:Node_Component_Black_Hole_Standard
File: objects.nodes.components.black_hole.standard.F90
Default value: 6.0
4. Input Parameters

**Description:** The factor by which the Bondi-Hoyle accretion rate of hot halo gas onto black holes is enhanced.

Name: `bondiHoyleAccretionEnhancementSpheroid`
Attached to: `module:Node_Component_Black_Hole_Standard`
File: `objects.nodes.components.black_hole.standard.F90`
Default value: 5
Description: The factor by which the Bondi-Hoyle accretion rate of spheroid gas onto black holes is enhanced.

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

Name: `burstCriticalGasFraction`
Attached to: `module:Satellite_Merging_Mass_Movements_Baugh2005`
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: `burstMassRatio`
Attached to: `module:Satellite_Merging_Mass_Movements_Baugh2005`
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: `chemicalReactionRateMethods`
Attached to: `module:Chemical_Reaction_Rates`
File: `chemical.reaction_rates.F90`
Default value: hydrogenNetwork
Description: The names of the methods to be used for computing chemical reaction rates.
Name: chemicalStateFile
Attached to: module:Chemical_States_CIE_File
File: 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: chemicalStateMethod
Attached to: module:Chemical_States
File: chemical.state.F90
Default value: atomicCIECloudy
Description: The name of the method to be used for computing the chemical state.

Name: chemicalsToTrack
Attached to: module:Chemical_Abundances_Structure
File: objects.chemical_abundances.F90
Default value: HASH(0x191c0bc0)
Description: The names of the chemicals to be tracked.

Name: conditionalStellarMassFunctionBehrooziAlphaSatellite
Attached to: module:Conditional_Stellar_Mass_Functions_Behroozi2010
Default value: 1.0 (Leauthaud et al. 2011; \(z_1\) sample using their SIG_MOD1 method)
Description: The parameter \(\alpha_{\text{sat}}\) from the fitting functions of Behroozi et al. [2010].

Name: conditionalStellarMassFunctionBehrooziBCut
Attached to: module:Conditional_Stellar_Mass_Functions_Behroozi2010
Default value: 1.47 (Leauthaud et al. 2011; \(z_1\) sample using their SIG_MOD1 method)
Description: The parameter \(B_{\text{cut}}\) from the fitting functions of Behroozi et al. [2010].

Name: conditionalStellarMassFunctionBehrooziBSatellite
Attached to: module:Conditional_Stellar_Mass_Functions_Behroozi2010
Default value: 10.62 (Leauthaud et al. 2011; \(z_1\) sample using their SIG_MOD1 method)
Description: The parameter \(B_{\text{sat}}\) from the fitting functions of Behroozi et al. [2010].

Name: conditionalStellarMassFunctionBehrooziBetaCut
Attached to: module:Conditional_Stellar_Mass_Functions_Behroozi2010
Default value: \(-0.13\) (Leauthaud et al. 2011; \(z_1\) sample using their SIG_MOD1 method)
Description: The parameter \(\beta_{\text{cut}}\) from the fitting functions of Behroozi et al. [2010].

Name: conditionalStellarMassFunctionBehrooziBetaSatellite
Attached to: module:Conditional_Stellar_Mass_Functions_Behroozi2010
Default value: 0.859 (Leauthaud et al. 2011; \(z_1\) sample using their SIG_MOD1 method)
Description: The parameter \(\beta_{\text{sat}}\) from the fitting functions of Behroozi et al. [2010].
4. Input Parameters

Name: conditionalStellarMassFunctionBehrooziBeta
Attached to: module:Conditional_Stellar_Mass_Functions_Behroozi2010
Default value: 0.457 (Leauthaud et al. 2011; z₁ sample using their SIG_MOD1 method)
Description: The parameter β from the fitting functions of Behroozi et al. [2010].

Name: conditionalStellarMassFunctionBehrooziDelta
Attached to: module:Conditional_Stellar_Mass_Functions_Behroozi2010
Default value: 0.5666 (Leauthaud et al. 2011; z₁ sample using their SIG_MOD1 method)
Description: The parameter δ from the fitting functions of Behroozi et al. [2010].

Name: conditionalStellarMassFunctionBehrooziGamma
Attached to: module:Conditional_Stellar_Mass_Functions_Behroozi2010
Default value: 1.53 (Leauthaud et al. 2011; z₁ sample using their SIG_MOD1 method)
Description: The parameter γ from the fitting functions of Behroozi et al. [2010].

Name: conditionalStellarMassFunctionBehrooziLog10M1
Attached to: module:Conditional_Stellar_Mass_Functions_Behroozi2010
Default value: 12.520 (Leauthaud et al. 2011; z₁ sample using their SIG_MOD1 method)
Description: The parameter log₁₀ M₁ from the fitting functions of Behroozi et al. [2010].

Name: conditionalStellarMassFunctionBehrooziLog10Mstar0
Attached to: module:Conditional_Stellar_Mass_Functions_Behroozi2010
Default value: 10.916 (Leauthaud et al. 2011; z₁ sample using their SIG_MOD1 method)
Description: The parameter log₁₀ M⋆,₀ from the fitting functions of Behroozi et al. [2010].

Name: conditionalStellarMassFunctionBehrooziSigmaLogMstar
Attached to: module:Conditional_Stellar_Mass_Functions_Behroozi2010
Default value: 0.206 (Leauthaud et al. 2011; z₁ sample using their SIG_MOD1 method)
Description: The parameter σ log M⋆ from the fitting functions of Behroozi et al. [2010].

Name: conditionalStellarMassFunctionMethod
Attached to: module:Conditional_Stellar_Mass_Functions
File: halo_model.conditional_stellar_mass_function.F90
Default value: Behroozi2010
Description: The name of the method to be used for empirical models of the conditional stellar mass function.

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

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

**Name:** coolingMeanAngularMomentumFrom  
**Attached to:** module:Cooling_Specific_Angular_Momenta_Constant_Rotation  
**File:** cooling.specific-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.
4. Input Parameters

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

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

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

Name: coolingRotationVelocityFrom
Attached to: module:Cooling_Specific_Angular_MomentaConstant_Rotation
File: 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: coolingSpecificAngularMomentumMethod
Attached to: module:Cooling_Specific_Angular_Momenta
File: 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: coolingTimeAvailableAgeFactor
Attached to: module:Cooling_Time_Available_White_Frenk
File: 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: coolingTimeAvailableMethod
Attached to: module:Cooling_Times_Available
File: 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: coolingTimeMethod
Attached to: module:Cooling_Times
File: cooling.cooling_time.F90
Default value: simple
Description: The name of the method to be use for computing cooling times.

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

Name: cosmologyMethod
Attached to: module:Cosmology_Functions
File: cosmology.functions.F90
Default value: matter-lambda
Description: The name of the method to be used for cosmology calculations.

Name: criticalOverdensityMassScalingMethod
Attached to: module:Critical_Overdensity
File: 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.
4. Input Parameters

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

Name: darkEnergyEquationOfStateW0
Attached to: module:Cosmology_Functions_Matter_Dark_Energy
File: cosmology.functions.matter_dark_energy.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: darkEnergyEquationOfStateW1
Attached to: module:Cosmology_Functions_Matter_Dark_Energy
File: cosmology.functions.matter_dark_energy.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: darkMatterAccretionHistoryMethod
Attached to: module:Dark_Matter_Halo_Mass_Accretion_Histories
File: 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: darkMatterConcentrationMethod
Attached to: module:Dark_Matter_Profiles_Concentrations
File: dark_matter_profiles.structure.concentration.F90
Default value: Gao2008
Description: The name of the method to be used for calculations of dark matter halo density profile concentrations.

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

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

Name: darkMatterProfileMethod
Attached to: module:Dark_Matter_Profiles
File: dark_matter_profiles.F90
Default value: NFW
Description: The name of the method to be used for calculations of dark matter halo density profiles.

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

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

Name: diskOutflowExponent
Attached to: module:Star_Formation_Feedback_Disks_Power_Law
File: 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: diskOutflowFraction
Attached to: module:Star_Formation_Feedback_Disks_Fixed
File: star_formation.feedback.disks.fixed.F90
4. Input Parameters

Default value: 0.01
Description: The ratio of outflow rate to star formation rate in disks.

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

Name: diskOutflowTimescaleMinimum
Attached to: module:Node_Component_Disk_Exponential
File: objects.nodes.components.disk.exponential.F90
Default value: $10^{-3}$
Description: The minimum timescale (in units of the disk dynamical time) on which outflows may deplete gas in the disk.

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

Name: diskOutputStarFormationRate
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: false
Description: Specifies whether the starFormationRate method of the exponential implementation of the disk component class should be output.

Name: diskRadiusSolverCole2000Method
Attached to: module:Node_Component_Disk_Exponential
File: objects.nodes.components.disk.exponential.F90
Default value: 1
Description: HASH(0x191d0050)

Name: diskStarFormationTimescaleMinimum
Attached to: module:Node_Component_Disk_Very_Simple
File: 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: diskStructureSolverRadius  
Attached to: module:Node_Component_Disk_Exponential  
File: 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: diskSuperwindMassLoading  
Attached to: module:Star_Formation_Expulsive_Feedback_Disks_Superwind  
File: star_formation.feedback_expulsion.disks.superwind.F90  
Default value: 2  
Description: The mass loading of the disk superwind.

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

Name: effectiveNumberNeutrinos  
Attached to: module:Transfer_Function_Eisenstein_Hu  
File: 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: elementsToTrack  
Attached to: module:Abundances_Structure  
File: objects.abundances.F90  
Default value: HASH(0x19151970)  
Description: The names of the elements to be tracked.

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

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

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

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

Name: excursionSetBarrierRemapScalingFactor
Attached to: module:Excursion_Sets_Barriers_Remap_Scale
File: 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: excursionSetFirstCrossingFarahiFileName
Attached to: module:Excursion_Sets_First_Crossing_Farahi
File: 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: excursionSetFirstCrossingFarahiFractionalTimeStep
Attached to: module:Excursion_Sets_First_Crossing_Farahi
File: 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: excursionSetFirstCrossingMethod
Attached to: module:Excursion_Sets_First_Crossings
File: 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: filterLightconeFixedTime
Attached to: module:Galacticus_Merger_Tree_Output_Filter_Lightcones
File: \textit{galacticus.output.merger\_tree.filters.lightcone.F90}
Default value: false
\textbf{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.

\textbf{Name: filterLightconeGeometryFileName}
\textbf{Attached to:} module:Galacticus\_Merger\_Tree\_Output\_Filter\_Lightcones
File: \textit{galacticus.output.merger\_tree.filters.lightcone.F90}
Default value: none
\textbf{Description:} The name of an XML file from which to read details of lightcone geometry.

\textbf{Name: freefallRadiusMethod}
\textbf{Attached to:} module:Freefall\_Radii
File: \textit{cooling.freefall_radius.F90}
Default value: darkMatterHalo
\textbf{Description:} The name of the method to be used for calculations of the freefall radius in cooling calculations.

\textbf{Name: freefallTimeAvailableMethod}
\textbf{Attached to:} module:Cooling\_Freefall\_Times\_Available
File: \textit{cooling.freefall_time_available.F90}
Default value: haloFormation
\textbf{Description:} The name of the method to be used when computing the time available for freefall in cooling calculations.

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

\textbf{Name: galacticStructureRadiusSolverInitialRadiusMethod}
\textbf{Attached to:} module:Galactic\_Structure\_Initial\_Radii
File: \textit{galactic\_structure.radius_solver.initial_radii.F90}
Default value: adiabatic
\textbf{Description:} Selects the method to be used to determine initial radii in the dark matter halo when solving for galactic structure.

\textbf{Name: galacticStructureRadiusSolverMethod}
\textbf{Attached to:} module:Galactic\_Structure\_Radii
File: \textit{galactic\_structure.radius_solver.F90}
Default value: adiabatic
\textbf{Description:} Selects the method to be used for solving for galactic structure.

\textbf{Name: galacticusOutputFileName}
\textbf{Attached to:} module:Galacticus\_Output\_Open
File: \textit{galacticus.output.HDF5.open.F90}
Default value: galacticus.hdf5
\textbf{Description:} The name of the file to which GALACTICUS results will be written.
4. Input Parameters

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

Name: generalizedPressSchechterDeltaStepMaximum
Attached to: module:Generalized_Press_Schechter_Branching
File: 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: generalizedPressSchechterMinimumMass
Attached to: module:Generalized_Press_Schechter_Branching
File: 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: generalizedPressSchechterSmoothAccretion
Attached to: module:Generalized_Press_Schechter_Branching
File: 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: haloMassFunctionMethod
Attached to: module:Halo_Mass_Function
File: 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: haloMassFunctionSamplingMethod
Attached to: module:Merger_Trees_Mass_Function_Sampling
File: 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: haloMassFunctionSamplingStellarMassFunctionErrorAlpha
Attached to: module:Merger_Trees_Mass_Function_Sampling_Stellar_MF
File: 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: haloMassFunctionSamplingStellarMassFunctionErrorBeta
Attached to: module:Merger_Trees_Mass_Function_Sampling_Stellar_MF
File: 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: haloMassFunctionSamplingStellarMassFunctionErrorConstant
Attached to: module:Merger_Trees_Mass_Function_Sampling_Stellar_MF
File: 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: haloMassFunctionSamplingStellarMassFunctionErrorLogBinWidth
Attached to: module:Merger_Trees_Mass_Function_Sampling_Stellar_MF
File: merger_trees.construct.mass_function_sampling.stellar_mass_function.F90
Default value: none
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: haloMassFunctionSamplingStellarMassFunctionErrorMassMaximum
Attached to: module:Merger_Trees_Mass_Function_Sampling_Stellar_MF
File: 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: haloMassFunctionSamplingStellarMassFunctionErrorMassMinimum
Attached to: module:Merger_Trees_Mass_Function_Sampling_Stellar_MF
File: 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: haloMassFunctionSamplingStellarMassFunctionErrorMstar
Attached to: module:Merger_Trees_Mass_Function_Sampling_Stellar_MF
File: 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: haloMassFunctionSamplingStellarMassFunctionErrorPhi0
Attached to: module:Merger_Trees_Mass_Function_Sampling_Stellar_MF
File: 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.
4. Input Parameters

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

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

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

Name: haloModelWavenumberMaximum
Attached to: module:Galacticus_Output_Halo_Models
File: 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: haloModelWavenumberMinimum
Attached to: module:Galacticus_Output_Halo_Models
File: 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: haloModelWavenumberPointsPerDecade
Attached to: module:Galacticus_Output_Halo_Models
File: 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: haloReformationMassFactor
Attached to: module:Node_Component_Formation_Times_Cole2000
Default value: 2.0
**Description:** Factor by which halo mass must have increased to trigger a new formation event.

**Name:** haloReformationOnPromotionOnly  
**Attached to:** module:Node_Component_Formation_Times_Cole2000  
**File:** objects.nodes.components.formation_times.Cole2000.F90  
**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:** haloSpinDistributionMethod  
**Attached to:** module:Halo_Spin_Distributions  
**File:** dark_matter_halos.spins.distributions.F90  
**Default value:** Bett2007  
**Description:** The name of the method to be use for computing halo spin distributions.

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

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

**Name:** heightToRadialScaleDiskBlitzRosolowsky  
**Attached to:** module:Star_Formation_Rate_Surface_Density_Disks_BR  
**File:** 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:** heightToRadialScaleDisk  
**Attached to:** module:Node_Component_Disk_Exponential  
**File:** 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:** hotHaloAngularMomentumLossFraction  
**Attached to:** module:Node_Component_Hot_Halo_Standard  
**File:** 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:** hotHaloCoolingFromNode  
**Attached to:** module:Node_Component_Hot_Halo_Standard  
**File:** objects.nodes.components.hot_halo.standard.F90
4. Input Parameters

**Default value:** currentNode  
**Description:** Specifies whether the angular momentum of cooling gas should be computed from the “current node” or the “formation node”.

**Name:** hotHaloCoredIsothermalCoreRadiiMethod  
**Attached to:** module:Hot_Halo_Density_Cored_Isothermal_Core_Radii  
**File:** hot_halo.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 hot halo profiles.

**Name:** hotHaloDensityMethod  
**Attached to:** module:Hot_Halo_Density_Profile  
**File:** hot_halo.density_profile.F90  
**Default value:** coredIsothermal  
**Description:** The name of the method to be used for calculations of the hot halo density profile.

**Name:** hotHaloExcessHeatDrivesOutflow  
**Attached to:** module:Node_Component_Hot_Halo_Standard  
**File:** 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:** hotHaloExpulsionRateMaximum  
**Attached to:** module:Node_Component_Hot_Halo_Standard  
**File:** 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:** hotHaloNodeMergerLimitBaryonFraction  
**Attached to:** module:Node_Component_Hot_Halo_Standard  
**File:** 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:** hotHaloOutflowAngularMomentumAlwaysGrows  
**Attached to:** module:Node_Component_Hot_Halo_Standard  
**File:** 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:** hotHaloOutflowReturnOnFormation  
**Attached to:** module:Node_Component_Hot_Halo_Standard
File: 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:** hotHaloOutflowReturnRate  
**Attached to:** module:Node_Component_Hot_Halo_Standard  
**File:** 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:** hotHaloOutflowStrippingEfficiency  
**Attached to:** module:Node_Component_Hot_Halo_Standard  
**File:** 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:** hotHaloRamPressureForceMethod  
**Attached to:** module:Hot_Halo_Ram_Pressure_Forces  
**File:** 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:** hotHaloRamPressureStrippingMethod  
**Attached to:** module:Hot_Halo_Ram_Pressure_Stripping  
**File:** 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:** hotHaloTemperatureMethod  
**Attached to:** module:Hot_Halo_Temperature_Profile  
**File:** hot_halo.temperature_profile.F90  
**Default value:** virial  
**Description:** The name of the method to be used for computing hot halo temperature profiles.

**Name:** hotHaloTrackStrippedGas  
**Attached to:** module:Node_Component_Hot_Halo_Standard  
**File:** 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:** hydrogenNetworkCMBOnly  
**Attached to:** module:Chemical_Hydrogen_Rates  
**File:** 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.
4. Input Parameters

Name: hydrogenNetworkFast
Attached to: module:Chemical_Hydrogen_Rates
File: 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, H\(^{2}\) reactions are ignored and other slow reactions are ignored (see Abel et al. 1997).

Name: imfBaugh2005TopHeavyRecycledInstantaneous
Attached to: module:Star_Formation_IMF_Baugh2005TopHeavy
File: star_formation.IMF.Baugh2005TopHeavy.F90
Default value: 0.91 [Baugh et al., 2005]
Description: The recycled fraction for the Baugh et al. [2005] top-heavy initial mass function (IMF) in the instantaneous recycling approximation.

Name: imfBaugh2005TopHeavyYieldInstantaneous
Attached to: module:Star_Formation_IMF_Baugh2005TopHeavy
File: star_formation.IMF.Baugh2005TopHeavy.F90
Default value: 0.15 [Baugh et al., 2005]
Description: The yield for the Baugh et al. [2005] top-heavy IMF in the instantaneous recycling approximation.

Name: imfChabrierRecycledInstantaneous
Attached to: module:Star_Formation_IMF_Chabrier
File: star_formation.IMF.Chabrier.F90
Default value: 0.46 (internally computed)
Description: The recycled fraction for the Chabrier IMF in the instantaneous recycling approximation.

Name: imfChabrierYieldInstantaneous
Attached to: module:Star_Formation_IMF_Chabrier
File: star_formation.IMF.Chabrier.F90
Default value: 0.035 (internally computed)
Description: The yield for the Chabrier IMF in the instantaneous recycling approximation.

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

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

Name: imfMillerScaloRecycledInstantaneous
Attached to: module:Star_Formation_IMF_MillerScalo
File: 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: imfMillerScaloYieldInstantaneous
Attached to: module:Star_Formation_IMF_MillerScalo
File: 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: imfPiecewisePowerLawExponents
Attached to: module:Star_Formation_IMF_PiecewisePowerLaw
File: 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: imfPiecewisePowerLawMassPoints
Attached to: module:Star_Formation_IMF_PiecewisePowerLaw
File: 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: imfPiecewisePowerLawRecycledInstantaneous
Attached to: module:Star_Formation_IMF_PiecewisePowerLaw
File: 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: imfPiecewisePowerLawYieldInstantaneous
Attached to: module:Star_Formation_IMF_PiecewisePowerLaw
File: 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.
4. Input Parameters

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

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

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

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

Name: imfSelectionSpheroid
Attached to: module:Star_Formation_IMF_Select_Disk_Spheroid
File: 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: infallRadiusMethod
Attached to: module:Cooling_Infall_Radii
File: 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: initialMassForSupernovaeTypeII
Attached to: module:Stellar_Feedback_Standard
File: stellar_astrophysics.feedback.standard.F90
Default value: 8
Description: The minimum mass that a star must have in order that is result in a Type II supernova.

Name: intergalaticMediumStateFileName
Attached to: module:Intergalactic_Medium_State_File
File: intergalatic_medium.state.file.F90
Default value: none
Description: The name of the file from which to read intergalactic medium state data.

Name: intergalaticMediumStateMethod
Attached to: module:Intergalactic_Medium_State
File: intergalatic_medium.state.F90
Default value: RecFast
Description: The name of the method to be used for computing the state of the intergalactic medium.

Name: isothermalCoreRadiusOverScaleRadius
Attached to: module:Hot_Halo_Density_Cored_Isothermal_Core_Radii_Growing_Core
File: hot_halo.density_profile.cored_isothermal.core_radius.growing_core.F90
Default value: 0.1
Description: The core radius in the “cored isothermal” hot halo density profile in units of the dark matter profile scale radius.

Name: isothermalCoreRadiusOverVirialRadiusMaximum
Attached to: module:Hot_Halo_Density_Cored_Isothermal_Core_Radii_Growing_Core
File: hot_halo.density_profile.cored_isothermal.core_radius.growing_core.F90
Default value: 10
Description: The maximum core radius in the “cored isothermal” hot halo density profile in units of the virial radius.

Name: isothermalCoreRadiusOverVirialRadius
Attached to: module:Hot_Halo_Density_Cored_Isothermal_Core_Radii_Virial_Fraction
File: hot_halo.density_profile.cored_isothermal.core_radius.virial_radius_fraction.F90
Default value: 0.3
Description: The core radius in the “cored isothermal” hot halo density profile in units of the virial radius.

Name: linearGrowthMethod
Attached to: module:Linear_Growth
4. Input Parameters

File: 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: lognormalSpinDistributionMedian
Attached to: module:Halo_Spin_Distributions_Lognormal
File: 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: lognormalSpinDistributionSigma
Attached to: module:Halo_Spin_Distributions_Lognormal
File: 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: luminosityBandRedshift
Attached to: module:Stellar_Population_Properties_Luminosities
File: stellar_populations.properties.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: luminosityFilterAbsoluteMagnitudeThresholds
Attached to: module:Galacticus_Merger_Tree_Output_Filter_Luminosities
File: 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: luminosityFilter
Attached to: module:Stellar_Population_Properties_Luminosities
File: stellar_populations.properties.luminosities.F90
Default value: none
Description: The filter name for each stellar luminosity to be computed.

Name: luminosityOutputOption
Attached to: module:Stellar_Population_Properties_Luminosities
File: stellar_populations.properties.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.

Name: luminosityPostprocessSet
Attached to: module:Stellar_Population_Properties_Luminosities
File: stellar_populations.properties.luminosities.F90
Default value: none
Description: The name of the set of postprocessing algorithms to apply to this filter.

Name: luminosityRedshift
Attached to: module:Stellar_Population_Properties_Luminosities
File: stellar_populations.properties.luminosities.F90
Default value: none
Description: The redshift for which to compute each specified stellar luminosity.

Name: luminosityType
Attached to: module:Stellar_Population_Properties_Luminosities
File: stellar_populations.properties.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: majorMergerMassRatio
Attached to: module:Satellite_Merging_Mass_Movements_Simple
File: 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: massAccretionHistoryOutput
Attached to: module:Merger_Tree_Mass_Accretion_History
File: 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: mergeTargetTimeOffsetMaximumAbsolute
Attached to: module:Merger_Tree_Timesteps_Satellite
File: 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: mergeTargetTimeOffsetMaximumRelative
Attached to: module:Merger_Tree_Timesteps_Satellite
File: 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: mergerRemnantRadiativeEfficiency
Attached to: module:Satellite_Merging_Remnant_Sizes_Covington2008

---

1 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.
4. Input Parameters

Default value: 2.75 [Covington et al., 2008]
Description: The coefficient, $C_{\text{rad}}$ energy used in the Covington et al. [2008] merger remnant size algorithm.

Name: mergerRemnantSizeOrbitalEnergy
Attached to: module:Satellite_Merging_Remnant_Sizes_Covington2008
Default value: 1
Description: The orbital energy used in the “Covington2008” merger remnant sizes calculation in units of the characteristic orbital energy.

Name: mergerTreeBaseRedshift
Attached to: module:Merger_Tree_Smooth_Accretion
File: 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: mergerTreeBuildCole2000AccretionLimit
Attached to: module:Merger_Tree_Builder_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: mergerTreeBuildCole2000HighestRedshift
Attached to: module:Merger_Tree_Builder_Cole2000
Default value: 10^5
Description: The highest redshift to which merger trees will be built in the Cole et al. [2000] method.

Name: mergerTreeBuildCole2000MassResolution
Attached to: module:Merger_Tree_Builder_Cole2000
Default value: $5 \times 10^9$
Description: The minimum mass (in units of $M_\odot$) of halos to be resolved in merger trees built using the Cole et al. [2000] method.

Name: mergerTreeBuildCole2000MergeProbability
Attached to: module:Merger_Tree_Builder_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: mergerTreeBuildHaloMassMaximum
Attached to: module:Merger_Tree_Builder
File: 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**: `mergerTreeBuildHaloMassMinimum`  
**Attached to**: `module:Merger_Tree_Build`  
**File**: `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**: `mergerTreeBuildMethod`  
**Attached to**: `module:Merger_Tree_Build`  
**File**: `merger_trees.construct.build.F90`  
**Default value**: `Cole2000`  
**Description**: The name of the method to be used to build merger trees.

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

**Name**: `mergerTreeBuildTreesBeginAtTree`  
**Attached to**: `module:Merger_Tree_Build`  
**File**: `merger_trees.construct.build.F90`  
**Default value**: `1`  
**Description**: The index (in order of increasing base halo mass) of the tree at which to begin when building merger trees.

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

**Name**: `mergerTreeBuildTreesHaloMassExponent`  
**Attached to**: `module:Merger_Trees_Mass_Function_Sampling_Power_Law`  
**File**: `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 $\alpha=\text{mergerTreeBuildTreesHaloMassExponent}$.
4. Input Parameters

Name: `mergerTreeBuildTreesHaloMassGaussianMean`
Attached to: module: `Merger_Trees_Mass_Function_Sampling_Gaussian`
File: `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: `mergerTreeBuildTreesHaloMassGaussianSigma`
Attached to: module: `Merger_Trees_Mass_Function_Sampling_Gaussian`
File: `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: `mergerTreeBuildTreesPerDecade`
Attached to: module: `Merger_Tree_Build`
File: `merger_trees.construct.build.F90`
Default value: 10
Description: The number of merger trees to build per decade of base halo mass.

Name: `mergerTreeBuildTreesProcessDescending`
Attached to: module: `Merger_Tree_Build`
File: `merger_trees.construct.build.F90`
Default value: true
Description: If true, causes merger trees to be processed in order of decreasing mass.

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

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

Name: `mergerTreeExportFileName`
Attached to: module: `Merger_Trees_Write`
File: `merger_trees.write.F90`
Default value: galacticusExportedTrees.hdf5
Description: The name of the file to which merger trees will be exported.
Name: mergerTreeExportOutputFormat
Attached to: module:Merger_Trees_Write
File: merger_trees.write.F90
Default value: galacticus
Description: The output format to use when exporting merger trees.

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

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

Name: mergerTreePruneHierarchyAtDepth
Attached to: module:Merger_Trees_Prune_Hierarchy
File: merger_trees.prune_hierarchy.F90

4. Input Parameters

- **mergerTreePruningMassThreshold**
  - **Default value:** 0
  - **Description:** The depth in the hierarchy at which to prune merger trees. (Zero indicates to not prune.)

- **mergerTreeReadAllowBranchJumps**
  - **Default value:** true
  - **Description:** Specifies whether nodes are allowed to jump between branches.

- **mergerTreeReadAllowSubhaloPromotions**
  - **Default value:** true
  - **Description:** Specifies whether subhalos are permitted to be promoted to being isolated halos.

- **mergerTreeReadBeginAt**
  - **Default value:** -1
  - **Description:** Specifies the index of the tree to begin at. (Use -1 to always begin with the first tree.)

- **mergerTreeReadFileName**
  - **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.

- **mergerTreeReadMismatchIsFatal**
  - **Default value:** true
  - **Description:** Specifies whether mismatches in cosmological parameter values between GALACTICUS and the merger tree file should be considered fatal.

- **mergerTreeReadMissingHostsAreFatal**
  - **Default value:** true
  - **Description:** Specifies whether nodes with missing host nodes should be considered to be fatal—see §6.2.1.

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

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

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

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

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

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

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

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

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

Name: `mergerTreeRegridEndExpansionFactor`
Attached to: `module:Merger_Trees_Regrid_Times`
File: `merger_trees.regrid_times.F90`
Default value: false
Description: Ending expansion factor to use when regridding merger trees.
Name: mergerTreeRegridSpacing
Attached to: module:Merger_Trees_Regrid_Times
File: merger_trees.regrid_times.F90
Default value: false
Description: Type of spacing to use in merger tree regridding (linear or logarithmic).

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

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

Name: mergerTreeStateStoreFile
Attached to: module:Merger_Trees_State_Store
File: 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: mergerTreeStructureOutputDarkMatterProfileScale
Attached to: module:Node_Component_Dark_Matter_PROFILE_Scale
File: 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: mergerTreeStructureOutputDarkMatterProfileShape
Attached to: module:Node_Component_Dark_Matter_PROFILE_Scale_Shape
File: 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: mergerTreeStructureOutputVirialQuantities
Attached to: module:Merger_Tree_Output_Structure
File: 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: mergerTreeStructureOutput
Attached to: module:Merger_Tree_Output_Structure
File: merger_trees.output_structure.F90
Default value: false
4. Input Parameters

Description: Specifies whether or not to output the structure of merger trees prior to evolution.

Name: mergerTreesDumpStructure
Attached to: module:Merger_Trees_Evolve
File: merger_trees.evolve.F90
Default value: false
Description: Specifies whether merger tree structure should be dumped to a DOT file.

Name: mergerTreesWrite
Attached to: module:Merger_Trees_Write
File: merger_trees.write.F90
Default value: false
Description: Specifies whether or not trees should be written to file.

Name: mergingTimescaleMultiplier
Attached to: module:Dynamical_Friction_Timescale_Utilities
File: satellites.merging.dynamical_friction.timescale.utilities.F90
Default value: 0.75
Description: A multiplier for the merging timescale in dynamical friction timescale calculations.

Name: metaCollectTimingData
Attached to: module:Galacticus_Meta_Tree_Timing
File: galacticus.meta.tree_timing.F90
Default value: false
Description: Specifies whether or not collect and output data on the time spent processing trees.

Name: metaProfileTimeStepMaximum
Attached to: module:Galacticus_Meta_Evolver_Profiler
File: galacticus.meta.evolver_profiler.F90
Default value: 10 Gyr
Description: The largest timestep to use in profiling ODE solver steps.

Name: metaProfileTimeStepMinimum
Attached to: module:Galacticus_Meta_Evolver_Profiler
File: galacticus.meta.evolver_profiler.F90
Default value: 10^{-6} Gyr
Description: The smallest timestep to use in profiling ODE solver steps.

Name: metaProfileTimeStepPointsPerDecade
Attached to: module:Galacticus_Meta_Evolver_Profiler
File: 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: minorMergerGasMovesTo
Attached to: module:Satellite_Merging_Mass_Movements_Simple
File: 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: modifiedPressSchechterFirstOrderAccuracy
Attached to: module:Modified_Press_Schechter_Branching
File: 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: modifiedPressSchechterG0
Attached to: module:Modified_Press_Schechter_Branching
File: merger_trees.branching_probability.modified_Press_Schechter.F90
Default value: 0.57
Description: The parameter $G_0$ appearing in the modified merger rate expression of Parkinson et al. [2008].

Name: modifiedPressSchechterGamma1
Attached to: module:Modified_Press_Schechter_Branching
File: merger_trees.branching_probability.modified_Press_Schechter.F90
Default value: 0.38
Description: The parameter $\gamma_1$ appearing in the modified merger rate expression of Parkinson et al. [2008].

Name: modifiedPressSchechterGamma2
Attached to: module:Modified_Press_Schechter_Branching
File: merger_trees.branching_probability.modified_Press_Schechter.F90
Default value: -0.01
Description: The parameter $\gamma_2$ appearing in the modified merger rate expression of Parkinson et al. [2008].

Name: molecularComplexClumpingFactorKMT09
Attached to: module:Star_Formation_Rate_Surface_Density_Disks_KMT09
File: star_formation.rate_surface_density.disks.KMT09.F90
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: molecularFractionFastKMT09
Attached to: module:Star_Formation_Rate_Surface_Density_Disks_KMT09
File: 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: nfw96ConcentrationC
Attached to: module:Dark_Matter_Profiles_Concentrations_NFW1996
File: dark_matter_profiles.structure.concentration.NFW.F90
Default value: 2000 Navarro et al. [1996]
Description: The parameter $f$ appearing in the halo concentration algorithm of Navarro et al. [1996].
4. Input Parameters

Name: nfw96ConcentrationF
Attached to: module:Dark_Matter_Profiles_Concentrations_NFW1996
File: dark_matter_profiles.structure.concentration.NFW.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: nodeFormationMassFraction
Attached to: module:Node_Component_Merging_Statistics_Standard
File: 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: nodeMajorMergerFraction
Attached to: module:Node_Component_Merging_Statistics_Recent
File: 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: nodeMergersMethod
Attached to: module:Merger_Trees_Evolve_Node
File: merger_trees.evolve.node.F90
Default value: singleLevelHierarchy
Description: Selects the method to be used for handling node merger events.

Name: nodePromotionIndexShift
Attached to: module:Node_Promotion_Index_Shifts
File: 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: nodeRecentMajorMergerFromInfall
Attached to: module:Node_Component_Merging_Statistics_Recent
File: 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: nodeRecentMajorMergerIntervalType
Attached to: module:Node_Component_Merging_Statistics_Recent
File: 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: nodeRecentMajorMergerInterval

74
Attached to: module:Node_Component_Merging_Statistics_Recent
File: 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 \( \text{nodeRecentMajorMergerIntervalType} = \text{absolute} \), or in units of the halodynamic time if \( \text{nodeRecentMajorMergerIntervalType} = \text{dynamical} \).

Name: noninstantHistoryTimesCount
Attached to: module:Stellar_Population_Properties_Noninstantaneous
File: stellar_populations.properties.noninstantaneous.F90
Default value: 10
Description: The number of times at which a galaxy’s stellar properties history is stored.

Name: odeAlgorithm
Attached to: module:Merger_Trees_Evolve_Node
File: merger_trees.evolve.node.F90
Default value: Runge-Kutta-Cash-Karp
Description: The algorithm to use in the ODE solver.

Name: odeToleranceAbsolute
Attached to: module:Merger_Trees_Evolve_Node
File: merger_trees.evolve.node.F90
Default value: 0.01
Description: The absolute tolerance used in solving differential equations for node evolution.

Name: odeToleranceRelative
Attached to: module:Merger_Trees_Evolve_Node
File: merger_trees.evolve.node.F90
Default value: 0.01
Description: The relative tolerance used in solving differential equations for node evolution.

Name: outputDensityContrastDataDarkOnly
Attached to: module:Galacticus_Output_Trees_Density_Contrasts
File: 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: outputDensityContrastData
Attached to: module:Galacticus_Output_Trees_Density_Contrasts
File: 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: outputDensityContrastHaloLoaded
Attached to: module:Galacticus_Output_Trees_Density_Contrasts
File: galacticus.output.merger_tree.density_contrasts.F90
Default value: HASH(0x190ce0d0)
Description: Specifies whether baryonic loading of the halo should be accounted for when outputting
4. Input Parameters

density contrast data.

Name: outputDensityContrastValues
Attached to: module:Galacticus_Output_Trees_Density_Contrasts
File: galacticus.output.merger_tree.density_contrasts.F90
Default value: none
Description: A list of density contrasts at which to output data.

Name: outputDescendentIndices
Attached to: module:Galacticus_Output_Trees_Descendents
File: 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: outputHalfLightData
Attached to: module:Galacticus_Output_Tree_Half_Light_Properties
File: 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: outputHaloModelData
Attached to: module:Galacticus_Output_Halo_Models
File: 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: outputHotHaloCoolingRadii
Attached to: module:Cooling_Radii
File: cooling.cooling_radius.F90
Default value: false
Description: Determines whether or not cooling radii are output.

Name: outputHotHaloCoolingRates
Attached to: module:Cooling_Rates
File: cooling.cooling_rate.F90
Default value: false
Description: Determines whether or not cooling rates and radii are output.

Name: outputMainBranchStatus
Attached to: module:Galacticus_Output_Trees_Main_Branch
File: 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: outputMassProfileData
Attached to: module:Galacticus_Output_Tree_Mass_Profiles
File: 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: outputMassProfileRadii
Attached to: module:Galacticus_Output_Tree_Mass_Profiles
File: galacticus.output.merger_tree.mass_profile.F90
Default value: none
Description: A list of radii at which to output the mass profile.

Name: outputMostMassiveProgenitor
Attached to: module:Galacticus_Output_Most_Massive_Progenitors
File: galacticus.output.merger_tree.most_massive_progenitor.F90
Default value: false
Description: Specifies whether or not most massive progenitor status should be output.

Name: outputRedshifts
Attached to: module:Galacticus_Output_Times
File: galacticus.output.times.F90
Default value: 0
Description: A list of (space-separated) redshifts at which GALACTICUS results should be output. Redshifts need not be in any particular order.

Name: outputRotationCurveData
Attached to: module:Galacticus_Output_Trees_Rotation_Curve
File: 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: outputRotationCurveRadii
Attached to: module:Galacticus_Output_Trees_Rotation_Curve
File: galacticus.output.merger_tree.rotation_curve.F90
Default value: none
Description: Specifies the radii at which the rotation curve will be output.

Name: outputSatellitePericenterData
Attached to: module:Galacticus_Output_Trees_Satellite_Pericenter
File: galacticus.output.merger_tree.satellite_pericenter.F90
Default value: false
Description: Specifies whether or not satellite orbital pericenter data (radius, velocity) should be included in the output.

Name: outputVelocityDispersionData
Attached to: module:Galacticus_Output_Trees_Velocity_Dispersion
File: 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.
4. Input Parameters

**Name:** outputVelocityDispersionRadii  
**Attached to:** module:Galacticus_Output_Trees_Velocity_Dispersion  
**File:** galacticus.output.merger_tree.velocity_dispersion.F90  
**Default value:** none  
**Description:** Specifies the radii at which the velocity dispersion will be output.

**Name:** outputVirialData  
**Attached to:** module:Galacticus_Output_Trees_Virial  
**File:** galacticus.output.merger_tree.virial.F90  
**Default value:** false  
**Description:** Specifies whether or not virial data (radius, velocity) should be included in the output.

**Name:** powerSpectraPointsPerDecade  
**Attached to:** module:Power_Spectrum_Tasks  
**File:** power_spectra.tasks.F90  
**Default value:** 10  
**Description:** The number of points per decade of wavenumber at which to tabulate power spectra.

**Name:** powerSpectraWavenumberMaximum  
**Attached to:** module:Power_Spectrum_Tasks  
**File:** power_spectra.tasks.F90  
**Default value:** $10^3$ Mpc$^{-1}$  
**Description:** The maximum wavenumber at which to tabulate power spectra.

**Name:** powerSpectraWavenumberMinimum  
**Attached to:** module:Power_Spectrum_Tasks  
**File:** power_spectra.tasks.F90  
**Default value:** $10^{-3}$ Mpc$^{-1}$  
**Description:** The minimum wavenumber at which to tabulate power spectra.

**Name:** powerSpectrumIndex  
**Attached to:** module:Primordial_Power_Spectrum_Power_Law  
**File:** structure_formation.power_spectrum.primordial.power_law.F90  
**Default value:** 0.9646 (Hinshaw et al. 2012; CMB+$H_0$+BAO)  
**Description:** The index of the power-law primordial power spectrum.

**Name:** powerSpectrumMethod  
**Attached to:** module:Primordial_Power_Spectra  
**File:** 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:** powerSpectrumNonlinearMethod  
**Attached to:** module:Power_Spectra_Nonlinear  
**File:** 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:** powerSpectrumReferenceWavenumber
Attached to: module:Primordial_Power_Spectrum_Power_Law
File: 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: powerSpectrumRunning
Attached to: module:Primordial_Power_Spectrum_Power_Law
File: 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: powerSpectrumWindowFunctionMethod
Attached to: module:Power_Spectrum_Window_Functions
File: 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: powerSpectrumWindowFunctionSharpKSpaceNormalization
Attached to: module:Power_Spectrum_Window_Functions_TH_KSS_Hybrid
File: 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: powerSpectrumWindowFunctionSharpKSpaceTopHatRadiiRatio
Attached to: module:Power_Spectrum_Window_Functions_TH_KSS_Hybrid
File: 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: prada2011ConcentrationAlpha
Attached to: module:Dark_Matter_Profiles_Concentrations_Prada2011
Default value: 6.948 Prada et al. [2011]
Description: The parameter \( \alpha \) appearing in the halo concentration algorithm of Prada et al. [2011].

Name: prada2011ConcentrationA
Attached to: module:Dark_Matter_Profiles_Concentrations_Prada2011
Default value: 2.881 Prada et al. [2011]
Description: The parameter \( A \) appearing in the halo concentration algorithm of Prada et al. [2011].
4. Input Parameters

Name: prada2011ConcentrationBeta
Attached to: module:Dark_Matter_Profiles_Concentrations_Prada2011
Default value: 7.386 Prada et al. [2011]
Description: The parameter $\beta$ appearing in the halo concentration algorithm of Prada et al. [2011].

Name: prada2011ConcentrationB
Attached to: module:Dark_Matter_Profiles_Concentrations_Prada2011
Default value: 1.257 Prada et al. [2011]
Description: The parameter $b$ appearing in the halo concentration algorithm of Prada et al. [2011].

Name: prada2011ConcentrationC0
Attached to: module:Dark_Matter_Profiles_Concentrations_Prada2011
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: prada2011ConcentrationC1
Attached to: module:Dark_Matter_Profiles_Concentrations_Prada2011
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: prada2011ConcentrationC
Attached to: module:Dark_Matter_Profiles_Concentrations_Prada2011
Default value: 1.022 Prada et al. [2011]
Description: The parameter $c$ appearing in the halo concentration algorithm of Prada et al. [2011].

Name: prada2011ConcentrationD
Attached to: module:Dark_Matter_Profiles_Concentrations_Prada2011
Default value: 0.060 Prada et al. [2011]
Description: The parameter $d$ appearing in the halo concentration algorithm of Prada et al. [2011].

Name: prada2011ConcentrationInverseSigma0
Attached to: module:Dark_Matter_Profiles_Concentrations_Prada2011
Default value: 1.047 Prada et al. [2011]
Description: The parameter $\sigma_0^{-1}$ appearing in the halo concentration algorithm of Prada et al. [2011].

Name: prada2011ConcentrationInverseSigma1
Attached to: module:Dark_Matter_Profiles_Concentrations_Prada2011
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: prada2011ConcentrationX0
Attached to: module:Dark_Matter_Profiles_Concentrations_Prada2011
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: prada2011ConcentrationX1
Attached to: module:Dark_Matter_Profiles_Concentrations_Prada2011
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: pressureCharacteristicBlitzRosolowsky
Attached to: module:Star_Formation_Rate_Surface_Density_Disks_BR
File: 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: pressureExponentBlitzRosolowsky
Attached to: module:Star_Formation_Rate_Surface_Density_Disks_BR
File: 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: profileOdeEvolver
Attached to: module:Merger_Trees_Evolve_Node
File: merger_trees.evolve.node.F90
Default value: false
Description: Specifies whether or not to profile the ODE evolver.

Name: radiationIGBFileName
Attached to: module:Radiation_IGB_File
File: radiation.intergalactic_background.file.F90
Default value: none
Description: The name of the file containing a tabulation of the radiation field.

Name: radiationIntergalacticBackgroundMethod
Attached to: module:Radiation_Intergalactic_Background
File: 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: ramPressureStrippingFormFactor
Attached to: module:Hot_Halo_Ram_Pressure_Stripping_Font2008
File: hot_halo.ram_pressure_stripping.Font2008.F90
Default value: 2
4. Input Parameters

**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:** ramPressureStrippingMassLossRateDiskSimpleFractionalRateMaximum
**Attached to:** module:Ram_Pressure_Stripping_Mass_Loss_Rate_Disks_Simple
**File:** 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.

**Name:** ramPressureStrippingMassLossRateDisksMethod
**Attached to:** module:Ram_Pressure_Stripping_Mass_Loss_Rate_Disks
**File:** 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:** randomSeed
**Attached to:** module:Pseudo_Random
**File:** numerical.random.F90
**Default value:** 219
**Description:** A seed value for the random number generator.

**Name:** randomSpinResetMassFactor
**Attached to:** module:Node_Component_Spin_Random
**File:** 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:** recentPopulationsTimeLimit
**Attached to:** module:Stellar_Population_Spectra_Postprocessing_Recent
**File:** stellar_populations.spectra.postprocess.recent.F90
**Default value:** $10^7$ years
**Description:** The maximum age of stellar populations to retain in the “recent” spectra postprocessing method.

**Name:** reionizationSuppressionOpticalDepth
**Attached to:** module:Accretion_Halos_Simple
**File:** accretion.halo.simple.F90
**Default value:** none
**Description:** The optical depth to electron scattering below which baryonic accretion is suppressed.

**Name:** reionizationSuppressionRedshift
**Attached to:** module:Accretion_Halos_Simple
**File:** accretion.halo.simple.F90
**Default value:** 9.97 (Hinshaw et al. 2012; CMB+$H_0$+BAO)
**Description:** The redshift below which baryonic accretion is suppressed.
Attached to: module: Accretion_Halos_Simple
File: accretion.halo.simple.F90
Default value: 35.0
Description: The velocity scale below which baryonic accretion is suppressed.

Name: satelliteMergingMassMovementsMethod
Attached to: module: Satellite_Merging_Mass_Movements
File: satellites.merging.mass_movements.F90
Default value: simple
Description: Selects the method to be used for deciding mass movements during satellite mergers.

Name: satelliteMergingMethod
Attached to: module: Satellite_Merging_Timescales
File: satellites.merging.timescale.F90
Default value: Jiang2008
Description: The name of the method to be used to compute satellite merging timescales.

Name: satelliteMergingRemnantProgenitorPropertiesMethod
Attached to: module: Satellite_Merging_Remnant_Sizes_Progenitors
File: 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: satelliteMergingRemnantSizeMethod
Attached to: module: Satellite_Merging_Remnant_Sizes
File: satellites.merging.remnant_sizes.F90
Default value: Covington2008
Description: The name of the method to be used for computing merger remnant sizes.

Name: satelliteOrbitResetOnHaloFormation
Attached to: module: Node_Component_Satellite_Very_Simple
File: 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: satelliteOrbitStoreOrbitalParameters
Attached to: module: Node_Component_Satellite_Standard
File: 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: satelliteOutputVirialOrbit
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: false
Description: Specifies whether the virialOrbit method of the standard implementation of the satellite
4. Input Parameters

component class should be output.

**Name:** sigma_8  
**Attached to:** module:Merger_Trees_Simple  
**File:** merger_trees.file_maker.simple.F90  
**Default value:** none  
**Description:** The fractional mass fluctuation in the linear density field at the present day in spheres of radius 8 Mpc/h.

**Name:** simpleRadiusSolverUseFormationHalo  
**Attached to:** module:Galactic_Structure_Radii_Simple  
**File:** 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.

**Name:** source  
**Attached to:** module:Merger_Trees_Simple  
**File:** merger_trees.file_maker.simple.F90  
**Default value:** none  
**Description:** The source of the merger trees.

**Name:** spheroidAngularMomentumAtScaleRadius  
**Attached to:** module:Node_Component_Spheroid_Standard  
**File:** 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:** spheroidEnergeticOutflowMassRate  
**Attached to:** module:Node_Component_Spheroid_Standard  
**File:** 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 $V_{\text{spheroid}}^2$.

**Name:** spheroidMassDistribution  
**Attached to:** module:Node_Component_Spheroid_Standard  
**File:** objects.nodes.components.spheroid.standard.F90  
**Default value:** hernquist  
**Description:** The type of mass distribution to use for the standard spheroid component.

**Name:** spheroidMassToleranceAbsolute  
**Attached to:** module:Node_Component_Spheroid_Standard  
**File:** 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: spheroidOutflowExponent
Attached to: module:Star_Formation_Feedback_Spheroids_Power_Law
File: star_formation.feedback.spheroids.power_law.F90
Default value: 3.5
Description: The velocity scaling of the SNe-driven outflow rate in spheroids.

Name: spheroidOutflowTimescaleMinimum
Attached to: module:Node_Component_Spheroid_Standard
File: 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: spheroidOutflowVelocity
Attached to: module:Star_Formation_Feedback_Spheroids_Power_Law
File: 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: spheroidOutputStarFormationRate
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: false
Description: Specifies whether the starFormationRate method of the standard implementation of the spheroid component class should be output.

Name: spheroidSersicIndex
Attached to: module:Node_Component_Spheroid_Standard
File: objects.nodes.components.spheroid.standard.F90
Default value: 4
Description: The Sérsic index to use for the spheroid component mass distribution.

Name: spheroidSuperwindMassLoading
Attached to: module:Star_Formation_Expulsive_Feedback_Spheroids_Superwind
File: star_formation.feedback.expulsion.spheroids.superwind.F90
Default value: 2
Description: The mass loading of the spheroid superwind.

Name: spheroidSuperwindVelocity
Attached to: module:Star_Formation_Expulsive_Feedback_Spheroids_Superwind
File: star_formation.feedback.expulsion.spheroids.superwind.F90
Default value: 200 km/s
Description: The velocity scale of the spheroid superwind.

Name: spinDistributionBett2007Alpha
Attached to: module:Halo_Spin_Distributions_Bett2007
Default value: 2.509 [Bett et al., 2007]
4. Input Parameters

**Description:** The dispersion in a lognormal halo spin distribution.

**Name:** spinDistributionBett2007Lambda0  
**Attached to:** module:Halo_Spin_Distributions_Bett2007  
**File:** dark_matter_halos.spins.distributions.Bett2007.F90  
**Default value:** 0.04326 [Bett et al., 2007]  
**Description:** The dispersion in a lognormal halo spin distribution.

**Name:** stabilityThresholdGaseous  
**Attached to:** module:Galactic_Dynamics_Bar_Instabilities_ELN  
**File:** 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:** stabilityThresholdStellar  
**Attached to:** module:Galactic_Dynamics_Bar_Instabilities_ELN  
**File:** 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:** starFormationDiskEfficiency  
**Attached to:** module:Star_Formation_Timescale_Disks_Dynamical_Time  
**File:** 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:** starFormationDiskMinimumTimescale  
**Attached to:** module:Star_Formation_Timescale_Disks_Dynamical_Time  
**File:** star_formation.timescales.disks.dynamical_time.F90  
**Default value:** $10^{-3}$ Gyr  
**Description:** The minimum timescale for star formation in disks.

**Name:** starFormationDiskTimescale  
**Attached to:** unknown  
**File:** star_formation.timescales.disks.Baugh2005.cpp  
**Default value:** 8.0  
**Description:** The timescale (in Gyr) for star formation in the Baugh et al. [2005] prescription.

**Name:** starFormationDiskVelocityExponent  
**Attached to:** unknown  
**File:** star_formation.timescales.disks.Baugh2005.cpp  
**Default value:** -3.0  
**Description:** The exponent for disk velocity in the Baugh et al. [2005] prescription for star formation in galactic disks.

**Name:** starFormationExpansionExponent  
**Attached to:** unknown  
**File:** star_formation.timescales.disks.Baugh2005.cpp  
**Default value:** 0.0
**Description:** The exponent for expansion factor in the Baugh et al. [2005] prescription for star formation in galactic disks.

**Name:** starFormationExpulsiveFeedbackDisksMethod  
**Attached to:** module:Star_Formation_Feedback_Expulsion_Disks  
**File:** 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:** starFormationExpulsiveFeedbackSpheroidsMethod  
**Attached to:** module:Star_Formation_Feedback_Expulsion_Spheroids  
**File:** 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:** starFormationExtendedSchmidtGasExponent  
**Attached to:** module:Star_Formation_Rate_Surface_Density_Disks_ExSchmidt  
**File:** 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:** starFormationExtendedSchmidtNormalization  
**Attached to:** module:Star_Formation_Rate_Surface_Density_Disks_ExSchmidt  
**File:** 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 $[M\odot\,yr^{-1}\,pc^{-2}]$.

**Name:** starFormationExtendedSchmidtStarExponent  
**Attached to:** module:Star_Formation_Rate_Surface_Density_Disks_ExSchmidt  
**File:** 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:** starFormationFeedbackDisksCreasy2012Beta0  
**Attached to:** module:Star_Formation_Feedback_Disks_Creasey2012  
**File:** 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:** starFormationFeedbackDisksCreasy2012Mu  
**Attached to:** module:Star_Formation_Feedback_Disks_Creasey2012  
**File:** 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:** starFormationFeedbackDisksCreasy2012Nu  
**Attached to:** module:Star_Formation_Feedback_Disks_Creasey2012  
**File:** star_formation.feedback.disks.Creasey2012.F90  
**Default value:** 0.16
4. Input Parameters

**Description**: The factor $\nu$ appearing in the Creasey et al. [2012] model for supernovae feedback.

**Name**: `starFormationFeedbackDisksMethod`
**Attached to**: module: `Star_Formation_Feedback_Disks`
**File**: `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**: `starFormationFeedbackSpheroidsMethod`
**Attached to**: module: `Star_Formation_Feedback_Spheroids`
**File**: `star_formation.feedback.spheroids.F90`
**Default value**: `powerLaw`

**Description**: The name of the method to be used for calculations of SNe feedback in spheroids.

**Name**: `starFormationFrequencyKMT09`
**Attached to**: module: `Star_Formation_Rate_Surface_Density_Disks_KMT09`
**File**: `star_formation.rate_surface_density.disks.KMT09.F90`
**Default value**: $0.385 \times 10^{-10}$ [Krumholz et al., 2009]

**Description**: The star formation frequency (in units of Gyr$^{-1}$) in the “Krumholz-McKee-Tumlinson” star formation timescale calculation.

**Name**: `starFormationFrequencyNormalizationBlitzRosolowsky`
**Attached to**: module: `Star_Formation_Rate_Surface_Density_Disks_BR`
**File**: `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**: `starFormationHistoriesMethod`
**Attached to**: module: `Galacticus_Output_Star_Formation_Histories`
**File**: `galacticus.output.merger_tree.star_formation.F90`
**Default value**: `null`

**Description**: The method to use for computing and outputting star formation histories.

**Name**: `starFormationHistoryFineTimeStep`
**Attached to**: module: `Star_Formation_Histories_Metallicity_Split`
**File**: `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**: `starFormationHistoryFineTime`
**Attached to**: module: `Star_Formation_Histories_Metallicity_Split`
**File**: `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**: `starFormationHistoryMetallicityCount`
**Attached to**: module: `Star_Formation_Histories_Metallicity_Split`
File: 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: starFormationHistoryMetallicityMaximum
Attached to: module:Star_Formation_Histories_Metallicity_Split
File: 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: starFormationHistoryMetallicityMinimum
Attached to: module:Star_Formation_Histories_Metallicity_Split
File: 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: starFormationHistoryTimeStep
Attached to: module:Star_Formation_Histories_Metallicity_Split
File: 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: starFormationImfInstantaneousApproximationEffectiveAge
Attached to: module:Star_Formation_IMF
File: star_formation.IMF.F90
Default value: 13.8 Gyr
Description: The effective age to use for computing SNeIa yield when using the instantaneous stellar evolution approximation.

Name: starFormationImfInstantaneousApproximationMassLongLived
Attached to: module:Star_Formation_IMF
File: 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: starFormationImfInstantaneousApproximation
Attached to: module:Star_Formation_IMF
File: star_formation.IMF.F90
Default value: false
Description: Option controlling whether stellar evolution should follow the instantaneous approximation.

Name: starFormationKennicuttSchmidtExponentTruncated
Attached to: module:Star_Formation_Rate_Surface_Density_Disks_KS
File: star_formation.rate_surface_density.disks.Kennicutt-Schmidt.F90
Default value: true
4. Input Parameters

**Description:** The exponent of the $\Sigma_{\text{gas}}/\Sigma_{\text{crit}}$ term used in truncating the Kennicutt-Schmidt star formation law.

**Name:** starFormationKennicuttSchmidtExponent
**Attached to:** module:Star_Formation_Rate_Surface_Density_Disks_KS
**File:** 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:** starFormationKennicuttSchmidtNormalization
**Attached to:** module:Star_Formation_Rate_Surface_Density_Disks_KS
**File:** 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:** starFormationKennicuttSchmidtTruncate
**Attached to:** module:Star_Formation_Rate_Surface_Density_Disks_KS
**File:** 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:** starFormationRateSurfaceDensityDisksMethod
**Attached to:** module:Star_Formation_Rate_Surface_Density_Disks
**File:** 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:** starFormationSpheroidEfficiency
**Attached to:** module:Star(Formation_Timescale_Spheroids_Dynamical_Time
**File:** 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:** starFormationSpheroidMinimumTimescale
**Attached to:** module:Star_Formation_Timescale_Spheroids_Dynamical_Time
**File:** star_formation.timescales.spheroids.dynamical_time.F90
**Default value:** $10^{-3}$ Gyr
**Description:** The minimum timescale for star formation in disks.

**Name:** starFormationSpheroidVelocityExponent
**Attached to:** module:Star_Formation_Timescale_Spheroids_Dynamical_Time
**File:** 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:** starFormationTimescaleDisksFixedTimescale
**Attached to:** module:Star_Formation_Timescale_Disks_Fixed
**File:** star_formation.timescales.disks.fixed.F90
**Default value:** 1 Gyr
**Description:** The timescale for star formation in the fixed timescale model for disks.

**Name:** starFormationTimescaleDisksHaloScalingRedshiftExponent
**Attached to:** module:Star_Formation_Timescale_Disks_Halo_Scaling
**File:** 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:** starFormationTimescaleDisksHaloScalingTimescale
**Attached to:** module:Star_Formation_Timescale_Disks_Halo_Scaling
**File:** 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:** starFormationTimescaleDisksHaloScalingVirialVelocityExponent
**Attached to:** module:Star_Formation_Timescale_Disks_Halo_Scaling
**File:** 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:** starFormationTimescaleDisksMethod
**Attached to:** module:Star_Formation_Timescales_Disks
**File:** 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:** starFormationTimescaleSpheroidsMethod
**Attached to:** module:Star_Formation_Timescales_Spheroids
**File:** 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:** starveSatellites
**Attached to:** module:Node_Component_Hot_Halo_Standard
**File:** 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:** stateFileRoot
**Attached to:** module:Galacticus_State
**File:** galacticus.state.F90
**Default value:** none
**Description:** The root name of files to which the internal state is written (to permit restarts).

**Name:** stateRetrieveFileRoot
**Attached to:** module:Galacticus_State
4. Input Parameters

File: `galacticus.state.F90`
Default value: none
Description: The root name of files to which the internal state is retrieved from (to restart).

Name: `stellarAstrophysicsMethod`
Attached to: `module:Stellar_Astrophysics`
File: `stellar_astrophysics.F90`
Default value: file
Description: The name of the method to be used for stellar astrophysics calculations.

Name: `stellarDensityChangeBinaryMotion`
Attached to: `module:Black_Hole_Binary_Separations_Standard`
File: `black_holes.binaries.separation_growth_rate.standard.F90`
Default value: true
Description: The change in density due to the black hole's motion.

Name: `stellarFeedbackMethod`
Attached to: `module:Stellar_Feedback`
File: `stellar_astrophysics.feedback.F90`
Default value: standard
Description: The method to use for computing aspects of stellar feedback.

Name: `stellarMassFilterThreshold`
Attached to: `module:Galacticus_Merger_Tree_Output_Filter_Stellar_Masses`
File: `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: `stellarPopulationLuminosityIntegrationToleranceRelative`
Attached to: `module:Stellar_Population_Luminosities`
File: `stellar_populations.luminosities.F90`
Default value: $10^{-3}$
Description: The relative tolerance used when integrating the flux of stellar populations through filters.

Name: `stellarPopulationPropertiesMethod`
Attached to: `module:Stellar_Population_Properties`
File: `stellar_populations.properties.F90`
Default value: instantaneous
Description: The method to use for computing properties of stellar populations.

Name: `stellarPopulationSpectraMethod`
Attached to: `module:Stellar_Population_Spectra`
File: `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: `stellarPopulationSpectraPostprocessDefaultMethods`
Attached to: `module:Stellar_Population_Spectra_Postprocess`
File: stellar_populations.spectra.postprocess.F90  
Default value: Meiksin2006  
Description: The name of methods to be used for post-processing of stellar population spectra.

Name: stellarPropertiesFile  
Attached to: module:Stellar_Astrophysics_File  
File: 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: stellarTracksFile  
Attached to: module:Stellar_Astrophysics_Tracks_File  
File: 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: stellarTracksMethod  
Attached to: module:Stellar_Astrophysics_Tracks  
File: stellar_astrophysics.tracks.F90  
Default value: file  
Description: The name of the method to be used for stellar tracks calculations.

Name: stellarWindsMethod  
Attached to: module:Stellar_Astrophysics_Winds  
File: stellar_astrophysics.winds.F90  
Default value: standard  
Description: The method to use for computing aspects of stellar winds.

Name: summedNeutrinoMasses  
Attached to: module:Transfer_Function_Eisenstein_Hu  
File: structure_formation.transfer_function.Eisenstein_Hu.F90  
Default value: 0  
Description: The summed mass (in electron volts) of all neutrino species.

Name: supernovaEnergy  
Attached to: module:Stellar_Feedback_Standard  
File: stellar_astrophysics.feedback.standard.F90  
Default value: $10^{51}$ ergs  
Description: The energy produced by a supernova (in ergs).

Name: supernovaeIaMethod  
Attached to: module:Supernovae_Type_Ia  
File: stellar_astrophysics.supernovae_type_Ia.F90  
Default value: Nagashima  
Description: The method to use for computing properties of Type Ia supernovae.

Name: supernovaePopIIIMethod  
Attached to: module:Supernovae_Population_III
4. Input Parameters

File: `stellar_astrophysics.supernovae_PopulationIII.F90`
Default value: Heger-Woosley2002
Description: The method to use for computing properties of Population III supernovae.

Name: `surfaceDensityCriticalBlitzRosolowsky`
Attached to: `module:Star_Formation_Rate_Surface_Density_Disks_BR`
File: `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: `surfaceDensityExponentBlitzRosolowsky`
Attached to: `module:Star_Formation_Rate_Surface_Density_Disks_BR`
File: `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: `timePerTreeFitFileName`
Attached to: `module:Galacticus_Meta_Compute_Times_File`
File: `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: `timePerTreeMethod`
Attached to: `module:Galacticus_Meta_Compute_Times`
File: `galacticus.meta.compute_times.F90`
Default value: file
Description: The name of the method to be used for computing the time per tree.

Name: `timestepHistoryBegin`
Attached to: `module:Merger_Tree_Timesteps_History`
File: `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: `timestepHistoryEnd`
Attached to: `module:Merger_Tree_Timesteps_History`
File: `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: `timestepHistorySteps`
Attached to: `module:Merger_Tree_Timesteps_History`
File: `merger_trees.evolve.timesteps.history.F90`
Default value: 30
Description: The number of steps (spaced logarithmically in cosmic time) at which to tabulate the volume averaged history of galaxies.
Name: timestepHostAbsolute
Attached to: module:Merger_Trees_Evolve
File: 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: timestepHostRelative
Attached to: module:Merger_Trees_Evolve
File: 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: timestepRecordEvolutionBegin
Attached to: module:Merger_Tree_Timesteps_Record_Evolution
File: 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: timestepRecordEvolutionEnd
Attached to: module:Merger_Tree_Timesteps_Record_Evolution
File: 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: timestepRecordEvolutionSteps
Attached to: module:Merger_Tree_Timesteps_Record_Evolution
File: 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: timestepRecordEvolution
Attached to: module:Merger_Tree_Timesteps_Record_Evolution
File: 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: timestepSimpleAbsolute
Attached to: module:Merger_Tree_Timesteps_Simple
File: 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: timestepSimpleRelative
4. Input Parameters

**Attached to:** module:Merger_Tree_Timesteps_Simple
**File:** 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:** toomreParameterCritical
**Attached to:** module:Star_Formation_Rate_Surface_Density_Disks_KS
**File:** 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:** transferFunctionFile
**Attached to:** module:Transfer_Functions_File
**File:** 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:** transferFunctionMethod
**Attached to:** module:Transfer_Functions
**File:** structure_formation.transfer_function.F90
**Default value:** Eisenstein-Hu1999
**Description:** The name of the method to be used for computing the transfer function.

**Name:** transferFunctionWDMFreeStreamingLength
**Attached to:** module:Transfer_Function_BBKS
**File:** structure_formation.transfer_function.BBKS.F90
**Default value:** 0
**Description:** The warm dark matter free streaming length (in Mpc).

**Name:** transferFunctionWdmCutOffScale
**Attached to:** module:Transfer_Function_Eisenstein_Hu
**File:** 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:** transferFunctionWdmEpsilon
**Attached to:** module:Transfer_Function_Eisenstein_Hu
**File:** 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., 2001].

**Name:** transferFunctionWdmEta
**Attached to:** module:Transfer_Function_Eisenstein_Hu
**File:** 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: transferFunctionWdmNu
Attached to: module:Transfer_Function_Eisenstein_Hu
File: 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: transferFunction
Attached to: module:Merger_Trees_Simple
File: merger_trees.file_maker.simple.F90
Default value: none
Description: The type of transfer function used.

Name: treeBranchingMethod
Attached to: module:Merger_Tree_Branching
File: 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: treeBranchingModifierMethod
Attached to: module:Merger_Tree_Branching_Modifiers
File: 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: treeEvolveLimitLoadAverage
Attached to: module:Galacticus_Tasks_Evolve_Tree
File: galacticus.tasks.evolve_tree.F90
Default value: false
Description: Specifies whether or not to limit the load average.

Name: treeEvolveLoadAverageMaximum
Attached to: module:Galacticus_Tasks_Evolve_Tree
File: galacticus.tasks.evolve_tree.F90
Default value: processorCount
Description: The maximum load average for which new trees will be processed.

Name: treeEvolveWorkerCount
Attached to: module:Galacticus_Tasks_Evolve_Tree
File: galacticus.tasks.evolve_tree.F90
Default value: 1
Description: The number of workers that will work on this calculation.

Name: treeEvolveWorkerNumber
Attached to: module:Galacticus_Tasks_Evolve_Tree
File: galacticus.tasks.evolve_tree.F90
4. Input Parameters

Default value: 1
Description: The number of this worker.

Name: treeNodeMethodBasic
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: standard
Description: Specifies the implementation to be used for the basic component of nodes.

Name: treeNodeMethodBlackHole
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: standard
Description: Specifies the implementation to be used for the blackHole component of nodes.

Name: treeNodeMethodDarkMatterProfile
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: scale
Description: Specifies the implementation to be used for the darkMatterProfile component of nodes.

Name: treeNodeMethodDisk
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: exponential
Description: Specifies the implementation to be used for the disk component of nodes.

Name: treeNodeMethodFormationTime
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: null
Description: Specifies the implementation to be used for the formationTime component of nodes.

Name: treeNodeMethodHotHalo
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: standard
Description: Specifies the implementation to be used for the hotHalo component of nodes.

Name: treeNodeMethodIndices
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: null
Description: Specifies the implementation to be used for the indices component of nodes.

Name: treeNodeMethodInterOutput
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: null
Description: Specifies the implementation to be used for the interOutput component of nodes.

Name: treeNodeMethodMergingStatistics
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: null
Description: Specifies the implementation to be used for the mergingStatistics component of nodes.

Name: treeNodeMethodPosition
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: null
Description: Specifies the implementation to be used for the position component of nodes.

Name: treeNodeMethodSatellite
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: standard
Description: Specifies the implementation to be used for the satellite component of nodes.

Name: treeNodeMethodSpheroid
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: standard
Description: Specifies the implementation to be used for the spheroid component of nodes.

Name: treeNodeMethodSpin
Attached to: unknown
File: work/build/objects.nodes.components.Inc
Default value: random
Description: Specifies the implementation to be used for the spin component of nodes.

Name: tripleBlackHoleInteraction
Attached to: module:Node_Component_Black_Hole_Standard
File: objects.nodes.components.black_hole.standard.F90
Default value: false
Description: Determines whether or not triple black hole interactions will be accounted for.

Name: velocityDispersionDiskGas
Attached to: module:Star_Formation_Rate_Surface_Density_Disks_KS
File: starformation.rate_surface_density.disks.Kennicutt-Schmidt.F90
Default value: 10 [Leroy et al., 2008]
Description: The velocity dispersion of gas in disks.

Name: verbosityLevel
Attached to: module:Galacticus_Tasks_Basic
File: galacticus.tasks.basic.F90
Default value: 1
Description: The level of verbosity for GALACTICUS (higher values give more verbosity).
4. Input Parameters

Name: virialDensityContrastFixedType
Attached to: module:Virial_Densities_Fixed
File: structure_formation.virial_density_contrast.fixed.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: virialDensityContrastFixed
Attached to: module:Virial_Densities_Fixed
File: structure_formation.virial_density_contrast.fixed.F90
Default value: 200
Description: The virial density contrast to use in the fixed value model.

Name: virialDensityContrastMethod
Attached to: module:Virial_Density_Contrast
File: structure_formation.virial_density_contrast.F90
Default value: sphericalTopHat
Description: Selects the method to be used for computing halo virial density contrasts.

Name: virialDensityContrastSphericalTopHatDarkEnergyFixEnergyAt
Attached to: module:Spherical_Collapse_Matter_Dark_Energy
File: 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: virialOrbitsFixedRadialVelocity
Attached to: module:Virial_Orbits_Fixed
File: 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: virialOrbitsFixedTangentialVelocity
Attached to: module:Virial_Orbits_Fixed
File: 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: virialOrbitsMethod
Attached to: module:Virial_Orbits
File: 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: warmDarkMatterCriticalOverdensityGX
Attached to: module:Critical_Overdensity_Mass_Scalings_WDM
File: 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: warmDarkMatterCriticalOverdensityMX
Attached to: module:Critical_Overdensity_Mass_Scalings_WDM
File: 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: warmDarkMatterCriticalOverdensityUseFittingFunction
Attached to: module:Critical_Overdensity_Mass_Scalings_WDM
File: 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: zeroCoolingRateAboveVelocity
Attached to: module:Cooling_Rates_White_Frenk
File: 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. 

\(<\text{treeCount}\>\) is the total number of merger trees present in a given output, and \(<\text{nodeCount}\>\) 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 GALACTI-
cus 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 (if the code is being 
maintained using MERCURIAL, otherwise a value of −1 is entered). 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 $M_\odot/\text{Gyr}/\text{Mpc}^3$).
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 \(M_\odot/\text{Mpc}^3\)).

Dimensionful 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.

mergerTree subgroups (optional) A set of mergerTree groups 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. Optional Outputs

Numerous other quantities can be optionally output. These are documented below:

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 massAccretionHistoryOutput=true. If requested, a new group massAccretionHistories will be made in the GALACTICUS output file. It will contain groups called 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\(^1\):

- **nodeIndex** The index of the node in the tree;
- **nodeMass** The mass of the node at this point in the tree (in \(M_\odot\)). The 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 virialDensityContrastMethod parameter.

Pre-Evolution Merger Trees

GALACTICUS can output the full structure of merger trees prior to any evolution. Merger tree structure can be requested by setting mergerTreeStructureOutput=true. Structures are written to a new group, mergerTreeStructures, in the GALACTICUS output file. This group will contain groups called mergerTreeN where N is the merger tree index. Each such group will contain the following datasets:

- **nodeIndex** The index of the node in the tree;
- **childNodeIndex** The index of this node’s first child node;
- **parentNodeIndex** The index of this node’s parent node;
- **siblingNodeIndex** The index of this node’s sibling node;
- **nodeTime** The time at this point in the tree (in Gyr);
- **nodeMass** The mass of the node at this point in the tree (in \(M_\odot\)). The 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 virialDensityContrastMethod parameter.

Additional, optional, datasets can be added by setting appropriate input parameters. Currently these include:

- **Virial quantities** If mergerTreeStructureOutputVirialQuantities=true then two additional datasets are included:
  - **nodeVirialRadius** The virial radius of the node (in Mpc);
  - **nodeVirialVelocity** The virial velocity of the node (in km/s);

---

\(^1\)“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

Dark matter scale radii If mergerTreeStructureOutputDarkMatterScaleRadius=true then an additional dataset is included:

darkMatterScaleRadius The scale radius of this node’s dark matter halo profile (in Mpc);

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

```
use 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
my $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}.
```

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. Perl Module for Data Extraction

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 \texttt{Galacticus::Baryons} module implements a hot gas fraction property called \texttt{hotHaloFraction} or \texttt{hotHaloFrac}. It has the following form:

```perl
package Baryons;
use PDL;
use Galacticus::HDF5;
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 \texttt{%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 \texttt{$model} structure along with the name of the requested property. The subroutine should then simply evaluate the requested property and store it in the appropriate location within \texttt{$model}. Note that the subroutine can request additional datasets be loaded (as happens above where \texttt{hotHaloMass} and \texttt{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: \texttt{Galacticus::HDF5}.
- **redshift** The redshift at which the galaxy exists. Provided by: \texttt{Galacticus::Time}.
- **time** The cosmic time (in Gyr) at which the galaxy exists. Provided by: \texttt{Galacticus::Time}.
- **expansionFactor** The expansion factor at which the galaxy exists. Provided by: \texttt{Galacticus::Time}.
- **stellarMass** The sum of disk and spheroid stellar masses. Provided by: \texttt{Galacticus::StellarMass}.
- **starFormationRate** The sum of disk and spheroid star formation rates. Provided by: \texttt{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: \texttt{Galacticus::HostNode}.
- **stellarMass** The sum of disk and spheroid stellar masses (or, whichever of these exist in the model). Provided by: \texttt{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.

^(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(\^[^\:]+):([^\:]+):([^\:]+):z\(\[d\]+\)=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 $+2.5\log_{10}(1 + z)$ 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).
5.2. Perl Module for Data Extraction

\texttt{^grasilFlux\[d\].+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).

\texttt{^grasilInfraredLuminosity} The total infrared (8–1000\,µm) luminosity of the galaxy as computed by the Grasil code (see §5.5 for a full description).

\texttt{^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).

\texttt{^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).

\texttt{flux850micronHayward} The flux of the galaxy at 850\,µm computed using the fitting formula of Hayward et al. [2010], specifically:

\[
\frac{S_{850\mu m}}{\text{Jy}} = A \left( \frac{\dot{M}_*}{100 M_\odot \text{Gyr}^{-1}} \right)^\alpha \left( \frac{R_{\text{dust}} M_{\text{metals, gas}}}{10^8 M_\odot} \right)^\beta,
\]

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'\}\equiv R_{\text{dust}}, \{'fitNormalization'\}\equiv A, \{'starFormationRateExponent'\}\equiv \alpha, \{'dustMassExponent'\}\equiv \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.

\texttt{(disk|spheroid)LymanContinuumLuminosity:z\[d\].+}$ The luminosity (in units of \(10^{50}\) photons/s) of the Lyman continuum radiation of the disk or spheroid component at the specified redshift. The rest-frame \(\text{Lyc}\) filter must have been computed in Galacticus to allow this luminosity to be computed. If the \(\text{Lyc}\) filter was computed with a non-default postprocessing chain then the name of the chain should be specified in \$dataBlock-\{'lymanContinuum'\}-\{'postProcessingChain'\'} Provided by: Galacticus::Lyc.

\texttt{agnLuminosity:(\[^:\]+):(\[^:\]+):(\[^:\]+:)??$} The luminosity of the AGN in the specified filter, frame and redshift (specified as the first, second and third elements of the “;” separated label) in units of the zero-point luminosity of the AB-magnitude system. (Note that, consistent with Galacticus’s definitions for continuum luminosities, observed frame luminosities do not include the \(1+z\) factor arising from the compression of photon frequencies due to redshifting. As such, when observed frame line luminosities are converted to observed fluxes an additional multiplicative factor of \(1+z\) must be included.) The bolometric luminosity is computed from the black hole rest mass accretion rate and radiative efficiency. An SED for an AGN of this bolometric luminosity is then computed using the model of Hopkins et al. [2007]. If the final \(\text{alpha}[0-9-\_\+\_]\+) option is provided, then the luminosity computed will be a broad band luminosity (in units of Watts) converted from the photon count rate in the filter assuming a spectrum of the form \(f_\nu \propto \nu^\alpha\), as is typically assumed in converting observed AGN X-ray count rates to luminosities. Provided by: Galacticus::AGNLuminosities.

\texttt{columnDensity(disk|Spheroid)??$} The column density of hydrogen (in units of \(\text{cm}^{-2}\)) 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.
5. Extracting and Analyzing Results

\(^{\text{peakSFR}}\) The peak star formation rate for each galaxy, measured from the \texttt{starFormationHistories} output group (see §16.4.3). The peak star formation rate reported is therefore that when averaged over the bins used by the star formation history output method (see §16.4.3). Provided by: \texttt{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: \texttt{Galacticus::LensingAmplification}.

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 \texttt{Galacticus::HaloModel} Perl module. To use this module, \texttt{Galacticus} must be run with \texttt{[outputHaloModelData]=true} (see §13.6) 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, \texttt{[dataHash]}, used for the \texttt{Galacticus::HDF5} module, and construct a PDL, \texttt{[selectedGalaxies]}, which contains the indices (not the node indices, but the positions within the PDL arrays read in from the \texttt{Galacticus} output file) of galaxies for which the clustering is to be computed. A power spectrum can then be computed using:

\[
(\texttt{[waveNumber],[linearPowerSpectrum],[galaxyPowerSpectrum]}) = \texttt{&HaloModel::Compute_Power_Spectrum}([\texttt{model}],[\texttt{selectedGalaxies}],[\texttt{space} => "redshift"]);
\]

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 \texttt{space} option is set to \texttt{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:

\[
(\texttt{[separations],[galaxyCorrelationFunction]}) = \texttt{&HaloModel::Compute_Correlation_Function}([\texttt{waveNumber}],[\texttt{galaxyPowerSpectrum}],[\texttt{separationMinimum}],[\texttt{separationMaximum}],[\texttt{separationPointsPerDecade}]);
\]

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 \texttt{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. \texttt{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

\[
\int_{M_{\text{min}}}^{M_{\text{max}}} n(M) \, dM = w_i,
\]

(5.2)
5.4. Postprocessing Scripts

where \( n(M) \) is the dark matter halo mass function and

\[
M_{\text{min}} = \sqrt{M_{i-1}M_i},
\]

\[
M_{\text{min}} = \sqrt{M_iM_{i+1}}.
\]

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:

\[
\phi_k = \sum_i \sum_{j=1}^{N_i} \begin{cases} 
  w_i & \text{if } L_k - \Delta k/2 < l_{i,j} \leq L_k + \Delta k/2 \\
  0 & \text{otherwise}.
\end{cases}
\]

Building Redshift Catalogs

The 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 Galacticus model with at least two outputs. The module will first check if Galacticus was run with lightcone output (see §13.7). 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 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 preceding 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 comovingDistance) by selecting at random from the available comoving volume. From this comoving distance a redshift and luminosity distance (properties redshift and 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 Galacticus::Survey module provides an 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 §16.4.3. 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 them².

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.

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;

²You can put these files where ever you want. Usually, we place them into aux/Grasil/.
$\text{dataset}->{'grasilOptions'}->{'wavelengthCount'}$ Specifies the number of wavelengths to use in calculating radiative transfer (i.e. the “nlf” parameter in GRASIL);

$\text{dataset}->{'grasilOptions'}->{'radialGridCount'}$ Specifies the number of radial grid cells to use in calculating radiative transfer (i.e. the “ndr” parameter in GRASIL).

$\text{dataset}->{'grasilOptions'}->{'recomputeSEDs}$ 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.

$\text{dataset}->{'grasilOptions'}->{'maxThreads}$ 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.

$\text{dataset}->{'grasilOptions'}->{'cpuLimit}$ 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 $\text{grasilSEDs}/Output<outputIndex>/mergerTree<treeIndex>/node<nodeIndex>/SED$ 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 $\text{wavelength}$ and $\text{inclination}$. 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 $\text{model}$. 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\(^3\).

### 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 $[\text{metaCollectTimingData}=\text{true}]$ will cause tree timing data to be recorded and output to the $\text{metaData}/\text{treeTiming}$ group. Three datasets are written to this group:

- $\text{treeMasses}$ Gives the base node masses of the recorded trees (in units of $M_\odot$);
- $\text{treeConstructTimes}$ Gives the time (in seconds) taken to construct each merger tree;
- $\text{treeEvolveTimes}$ Gives the time (in seconds) taken to evolve each merger tree.

\(^3\)The selection element of $\text{model}$ 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. Extracting and Analyzing Results

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 §16.4.3). 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 reponse 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

```bash
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 files. 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 §16.4.1) 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):

1If you do not have a username and password for the Millennium Simulation database you can request one from contact@g-vo.org.
6.2. Merger Trees

<table>
<thead>
<tr>
<th>node</th>
<th>descendantNode</th>
<th>hostNode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

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 §11) 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 \( \text{mergerTreeReadMissingHostsAreFatal} = \text{false} \) 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 \( \text{mergerTreeReadAllowSubhaloPromotions} = \text{true} \) then such cases will be handled correctly (i.e. the subhalo will be promoted back to being an isolated halo). If \( \text{mergerTreeReadAllowSubhaloPromotions} = \text{false} \) 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 descendantNode 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 descendents 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 ($\Omega_M, \Omega_\Lambda, \Omega_b, H_0, \sigma_8$), 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 §11.9).

\(^2\)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.
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 angular Momentum dataset is available within the 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 §11.8).
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:

```xml
<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:

```xml
<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>
```

125
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 -->
```
7.1. Running GALACTICUS on N-body Merger Trees

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 \texttt{mergerTreeReadPresetOrbitsSetAll} option, if true, will cause such orbits to be assigned randomly using the selected XXXXX method, such that all orbits are assigned. The \texttt{mergerTreeReadPresetOrbitsAssertAllSet} option requires that all orbits be set—if \texttt{mergerTreeReadPresetOrbitsSetAll}=false and \texttt{mergerTreeReadPresetOrbitsAssertAllSet}=true then \texttt{Galacticus} will exit with an error message if any orbit cannot be set.

\section*{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

\begin{parameter}
  <name>treeNodeMethodSatelliteOrbit</name>
  <value>preset</value>
</parameter>
\begin{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,

\begin{parameter}
  <name>treeNodeMethodSatelliteOrbit</name>
  <value>standard</value>
</parameter>
\begin{parameter}
  <name>mergerTreeReadPresetMergerNodes</name>
  <value>false</value>
</parameter>
\begin{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, \texttt{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

\begin{parameter}
  <name>mergerTreeReadPresetMergerNodes</name>
</parameter>
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.

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>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.

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:

```xml
<parameter>
  <name>mergerTreeReadPresetSpins</name>
  <value>true</value>
</parameter>
```

The spin parameter is set using the angular momentum of each node stored in the merger tree file using:

\[
\lambda = \frac{|\mathbf{J}| |E|^{1/2}}{G M^{5/2}}
\]  

(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.

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:

```xml
<parameter>
  <name>mergerTreeReadPresetScaleRadii</name>
  <value>true</value>
</parameter>
```

\(^1\)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.
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 \texttt{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 whatever method is selected by the \texttt{darkMatterConcentrationMethod} parameter (see §12.7.3).

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.

\textbf{“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:

\begin{verbatim}
<parameter>
  <name>mergerTreeReadMissingHostsAreFatal</name>
  <value>false</value>
</parameter>
\end{verbatim}

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.

\subsection*{7.1.2. Analyzing the Output}

\textbf{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).

\textbf{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.

\subsection*{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 \textsc{Galacticus} involves first extracting the halos (and their merger trees) within this “lightcone” from a suitable N-body simulation, and then processing them through
7.2. Generating Mock Catalogs with Lightcones from the Millennium Simulation

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 -->
<parameter>
  <name>mergerTreeOutputFilters</name>
  <value>lightcone</value>
</parameter>

<parameter>
  <name>filterLightconeGeometryFileName</name>
  <value>lightconeDirectory/geometry.xml</value>
</parameter>

2Note that these are not the ID numbers of the trees, just a sequential count of all trees retrieved.
7. Tutorials

where the geometry.xml file will have been automatically created within the <lightconeDirectory> directory.

Selecting the lightcone filter automatically causes lightcone coordinate information (i.e. the position and velocity of each galaxy in a coordinate system with axes aligned along the line of sight of the lightcone and parallel to the two edges of the square field of view) to be output (see §13.7). The redshift of each galaxy is also output.

7.3. Using the Instantaneous Recycling Approximation

Choosing [stellarPopulationPropertiesMethod]=instantaneous will cause GALACTICUS to use the instantaneous recycling approximation for all calculations of stellar populations. The recycling rate and yield to use are set by the [imfNAMEReycledInstantaneous] and [imfNAMEYieldInstantaneous] parameters respectively, where NAME is the name of the appropriate IMF.

Setting [stellarPopulationPropertiesMethod]=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:

```xml
<parameter>
  <!-- Force the calculation of recycling, yields etc. to -->
  <!-- be done assuming instantaneous recycling -->
  <name>starFormationImfInstantaneousApproximation</name>
  <value>true</value>
</parameter>
```

```xml
<parameter>
  <!-- Set the mass of stars which should be used as the -->
  <!-- dividing line between long-lived and instantaneously -->
  <!-- evolving in this approximation. -->
  <name>starFormationImfInstantaneousApproximationMassLongLived</name>
  <value>1.0</value>
</parameter>
```

```xml
<parameter>
  <!-- Set the effective age of populations to use in this -->
  <!-- approximation when computing SNe numbers. -->
  <name>starFormationImfInstantaneousApproximationEffectiveAge</name>
  <value>13.8</value>
</parameter>
```

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

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 \[ \text{starFormationHistoryTimeStep} \] to give improved resolution in the star formation history for recently formed stars.

starFormationHistoryFineTime The period before each output for which the \[ \text{starFormationHistoryFineTimeStep} \] 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 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";
```

\(^3\)GRASIL SEDs are stored in the 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.
# 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 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 recomputeSED option to 1 instead. Fluxes are returned in units of Janskys, while the total infrared luminosity (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:

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 galacticus.hdf5 file. The SED will be shown for an inclination of 43.2° and the plot will be written to SED.pdf.

---

1Exercise 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.
8. Numerical Implementation

8.1. 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.

8.1.1. 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

$$\Delta t \leq t_{\text{parent}} - t \quad (8.1)$$

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

$$\Delta t \leq t_{\text{host}}, \quad (8.2)$$

$$\Delta t \leq \epsilon_{\text{host}}(a/\dot{a}), \quad (8.3)$$

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”.
8. Numerical Implementation

“Satellite” Criteria

For nodes which host satellite nodes, the “satellite” criterion asserts that

\[ \Delta t \leq \min(t_{\text{satellite}}) - t, \]  

(8.4)

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

\[ \Delta t \leq \min(t_{\text{sibling}}) - t, \]  

(8.5)

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

\[ \Delta t \leq \min(t_{\text{merge}}) - t, \]  

(8.6)

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.

8.1.2. General Criteria

“Simple” Criteria

The “simple” timestep criteria assert that

\[ \Delta t \leq t_{\text{simple}}, \]  

\[ \Delta t \leq \epsilon_{\text{simple}}(a/\dot{a}), \]  

(8.7) \hspace{1cm} (8.8)

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

\[ \Delta t \leq \Delta t_{\text{merge}}, \]  

(8.9)

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.
8.1. Timestepping Criteria

If the merge target has not been advanced to at least $t_{\text{required}}$ then instead
\[
\Delta t \leq \max(\Delta t_{\text{merge}} - \delta t_{\text{merge, maximum}}/2, 0),
\]
(8.10)
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.

8.1.3. Output Criteria

"History" Criteria

The “history” timestep criterion asserts that
\[
\Delta t \leq t_{\text{history}, i} - t
\]
(8.11)
where $t$ is the current time, $t_{\text{history}, i}$ is the $i^{\text{th}}$ 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
\[
\Delta t \leq t_{\text{record}, i} - t
\]
(8.12)
where $t$ is the current time, $t_{\text{record}, i}$ is the $i^{\text{th}}$ 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
9. Constraining \textsc{Galacticus}

9.1. Optimal Halo Mass Function Sampling

Suppose we want to fit parameters of the \textsc{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. \textsc{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.


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 \( M + M \, d \ln M \) per unit volume;

\( \gamma(M) \, d \ln M \) is the number of trees that we will simulate in the range \( M \) to \( M + M \, d \ln M \);

\( \alpha(M_*) \) 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 \( M \);

\( t(M) \) is the CPU time it takes to simulate a tree of mass \( M \).

To clarify, \( P(N|M_*, M; \delta \ln M_*) \) is the probability\(^1\) 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:

\[
\phi(M_*, M) \, \delta \ln M_* = \sum_{N=0}^{\infty} N P(N|M_*, M; \delta \ln M_*)
\]

(9.1)

The model estimate of the stellar mass function \( \Phi(M_*) \) (defined per unit \( \ln M_* \)) is

\[
\Phi(M_*) = \int_{0}^{\infty} \phi(M_*, M) \frac{n(M)}{\gamma(M)} \, d \ln M,
\]

(9.2)

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.

\(^1\)To put it another way, \( P(N|M_*, M; \delta \ln M_*) \) is closely related to the commonly used Halo Occupation Distribution.
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, \( \sigma^2(M_*) \), 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

\[
\ln L = -\frac{1}{2} \sum_i \frac{[\phi_{\text{obs},i} - \phi_i]^2}{\alpha_i^2 + \sigma_i^2},
\]

where the sum is taken over all data points, \( i \), and \( \sigma_i^2 \) is the variance in the model estimate and is given by

\[
\sigma^2(M_*) = \langle [\phi(M_*) - \phi(M_*)]^2 \rangle,
\]

where \( \phi(M_*) \) is the realization from a single model and \( \phi(M_*) \) 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,

\[
\sigma^2(M_*) = \sum_i \zeta_i^2(M_*; M)
\]

where \( \zeta_i^2(M_*; M) \) is the variance in the contribution to the stellar mass function from tree \( i \). This in turn is given by

\[
\zeta_i^2(M_*; M) = \psi^2(M_*; M) \left[ \frac{n(M)}{\gamma(M)} \right]^2,
\]

where \( \psi^2(M_*; M) \) is the variance in the conditional stellar mass function. In the continuum limit this becomes

\[
\sigma^2(M_*) = \int_0^\infty \psi^2(M_*; M) \left[ \frac{n(M)}{\gamma(M)} \right]^2 \gamma(M) d\ln M.
\]

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:

\[
\Delta \ln L = \sum_i \frac{[\phi_{\text{obs},i} - \phi_i]^2}{\alpha_i^2} - \frac{[\phi_{\text{obs},i} - \phi_i]^2}{\alpha_i^2 + \sigma_i^2}
\]

Of course, we don’t know the model prediction, \( \phi_i \), in advance\(^2\). 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\(^3\)

\[
F[\gamma(M)] = \sum_i \frac{\alpha_i^2}{\alpha_i^2 + \sigma_i^2}.
\]

If the bins all have the same \( \delta \ln M_* \) we can turn the sum into an integral

\[
F[\gamma(M)] = \int_0^\infty \frac{\alpha(M_*)^2}{\alpha(M_*)^2 + \sigma(M_*)^2} d\ln M_*.
\]

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

\[
\tau = \int_0^\infty t(M) \gamma(M) d\ln M.
\]

\(^2\)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.

\(^3\)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.
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

$$F[\gamma(M)] = \int_0^\infty \frac{\alpha(M_s)^2}{\alpha(M_s)^2 + \sigma(M_s)^2} d\ln M_s + \int_0^\infty \lambda(M) t(M)d\ln M.$$  \hspace{1cm} (9.12)

Finding the functional derivative and setting it equal to zero gives:

$$\gamma(M) = \sqrt{\frac{\xi(M)}{M(M)}}.$$  \hspace{1cm} (9.13)

in the limit where $\sigma(M_s) \ll \alpha(M_s)$, and where

$$\xi(M) = n^2(M) \int_{-\infty}^\infty \frac{\psi^2(M_s; M)}{\alpha^2(M_s)} d\ln M_s.$$  \hspace{1cm} (9.14)

The values of $\lambda$ and $\delta \ln M_s$, 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 9.2 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\(^5\):

$$\langle N_c(M_s|M) \rangle \equiv \int_{M_*}^{\infty} \phi_c(M'_s)d\ln M'_s = \frac{1}{2} \left[ 1 - \text{erf} \left( \frac{\log_{10} M_s - \log_{10} f_{\text{SHMR}}(M)}{\sqrt{2}\sigma_{\log M_s}} \right) \right].$$  \hspace{1cm} (9.15)

Here, the function $f_{\text{SHMR}}(M)$ is the solution of

$$\log_{10} M = \log_{10} M_1 + \beta \log_{10} \left( \frac{M_s}{M_{*,0}} \right) + \frac{(M_s/M_{*,0})^\delta}{1 + (M_s/M_{*,0})^{-\gamma}} - 1/2.$$  \hspace{1cm} (9.16)

For satellites,

$$\langle N_s(M_s|M) \rangle \equiv \int_{M_*}^{\infty} \phi_s(M'_s)d\ln M'_s = \langle N_c(M_s|M) \rangle \left( \frac{f_{\text{SHMR}}^{-1}(M_s)}{M_{\text{sat}}} \right)^{\alpha_{\text{sat}}} \exp \left( -\frac{M_{\text{cut}}}{f_{\text{SHMR}}^{-1}(M_s)} \right),$$  \hspace{1cm} (9.17)

where

$$\frac{M_{\text{sat}}}{10^{12}M_\odot} = B_{\text{sat}} \left( \frac{f_{\text{SHMR}}^{-1}(M_s)}{10^{12}M_\odot} \right)^{\beta_{\text{sat}}} ,$$  \hspace{1cm} (9.18)

and

$$\frac{M_{\text{cut}}}{10^{12}M_\odot} = B_{\text{cut}} \left( \frac{f_{\text{SHMR}}^{-1}(M_s)}{10^{12}M_\odot} \right)^{\beta_{\text{cut}}} .$$  \hspace{1cm} (9.19)

We use the best fit parameters from the \texttt{SIGMOD1} 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 9.1.

We assume that $P_c(N|M_s, M; \delta \ln M_s)$ is a Poisson distribution while $P_s(N|M_s, M; \delta \ln M_s)$ has a Bernoulli distribution, with each distribution’s free parameter fixed by the constraint of eqn. (9.1), and the assumed forms for $\phi_c$ and $\phi_s$.

\(^4\)This is the limit in which we would like our results to be.
\(^5\)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.
9. Constraining GALACTICUS

Table 9.1.: 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>0.566</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>

The errors in the Li and White [2009] observed stellar mass function are well fit by (see Fig. 9.1):

$$
\alpha(M_*) = 10^{-3} \left( \frac{M_*}{4.5 \times 10^{10} M_\odot} \right)^{-0.3} \exp \left(-\frac{M_*}{4.5 \times 10^{10} M_\odot}\right) + 10^{-7},
$$

and the tree processing time in GALACTICUS can be described by:

$$
\log_{10} t(M) = \sum_{i=0}^{2} C_i [\log_{10} M]^i
$$

with $C_0 = -0.73$, $C_1 = -0.20$ and $C_2 = 0.035$.

The resulting optimal sampling density curve is shown in Fig. 9.2 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.

### 9.1.2. 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.
9.1. Optimal Halo Mass Function Sampling

Figure 9.1.: Errors on the Li and White [2009] stellar mass function (points) and the fitting function (line) given by eqn. (9.20).
Figure 9.2.: 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.
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.
10. Definitions and Conventions Used in \textsc{GALACTICUS}

\textsc{GALACTICUS} adopts various definitions and conventions internally. These are explained below.

10.1. Luminosity Units

Galaxy luminosities are output in the \textit{AB magnitude} system, such that a luminosity of 1 corresponds to an object of $0^{\text{th}}$ absolute magnitude in the \textit{AB magnitude} system. This implies that the luminosities are in units of $4.4659 \times 10^{13}$ W/Hz.

10.2. Peculiar Velocities

Velocities in \textsc{GALACTICUS} are always \textit{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 \textsc{GALACTICUS}. Defining $t$ to be physical time and $x$ to be comoving position, \textsc{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

$$v_{\text{pec}} = \frac{dr}{dt} - Hr = a \frac{dx}{dt} = \frac{dx}{d\eta},$$ \hspace{1cm} (10.1)

where $d\eta = dt/a$ is conformal time.
11. 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).

11.1. (Supermassive) Black Hole

11.1.1. “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:

\[
\dot{M}_\bullet = (1 - \epsilon_{\text{radiation}} - \epsilon_{\text{jet}}) \dot{M}_0 \\
\dot{j}_\bullet = \dot{j}(M_\bullet, j_\bullet, \dot{M}_0),
\]

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 §16.4.1). 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 §12.19) 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:

\[
\dot{M}_{\text{hot halo, maximum}} = M_{\text{hot}} / \tau_{\text{sound crossing}},
\]

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.
11. Node Components

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:

\[
 f_{\text{hot}} = \begin{cases} 
 1 & \text{if } x < 0.9 \\
 y(x)^2[2y(x) - 3] + 1 & \text{if } 0.9 \leq x \leq 1.0 \\
 0 & \text{if } x > 1.0, 
\end{cases}
\]  
(11.4)

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 §12.3) 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 \([\text{Ostriker et al., 2010}]\)

\[
 L_{\text{wind}} = \epsilon_{\text{wind}}H(\epsilon_{\text{radiation}}, 1, s)\dot{M}_0c^2,
\]  
(11.5)

where \(\epsilon_{\text{wind}} = [\text{blackHoleWindEfficiency}]\) is the black hole wind efficiency, \(s=\text{blackHoleWindEfficiencyScalesWithRadiativeEfficiency}\), and

\[
 H(a, b, c) = \begin{cases} 
 a & \text{if } c = \text{true} \\
 b & \text{if } c = \text{false}, 
\end{cases}
\]  
(11.6)

is added to the gas component of the spheroid (which, presumably, will respond with an outflow for example—see §11.4 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.

\[
 P_{\text{wind}} < P_{\text{ISM}}/2m_H. 
\]  
(11.7)

Since \(\Omega^2\rho_{\text{wind}}V_{\text{wind}}^3 = L_{\text{wind}}\) 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

\[
 \rho_{\text{wind,critical}} = \frac{2m_HL_{\text{wind}}}{3\Omega r^2 V_{\text{wind}}^3 k_B T_{\text{ISM}}}. 
\]  
(11.8)

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

\[
 \langle \rho_{\text{ISM}} \rangle = \frac{3M_{\text{gas,spheroid}} r_{\text{spheroid}}^3}{4\pi}. 
\]  
(11.9)

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

\[
 f_{\text{wind}} = \begin{cases} 
 0 & \text{if } x < 0, \\
 3x^2 - 2x^3 & \text{if } 0 \leq x \leq 1, \\
 1 & \text{if } x > 1, 
\end{cases}
\]  
(11.10)

\(^1\)Specifically, the jet power multiplied by \(f_{\text{hot}}[\dot{M}_{\text{hot}}/\dot{M}_{\text{total}}](\Omega_M/\Omega_b)^2\) 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.
11.1. (Supermassive) Black Hole

where \( x = \rho_{\text{ISM}} / \rho_{\text{wind,critical}} - 1/2 \).

The radial position, \( r_\bullet \), evolves according to the selected radial migration method (see §16.4.1).

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

\[
a_h = \frac{G(M_{\bullet,1} + M_{\bullet,2})}{4\sigma^2}
\]

(11.11)

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

\[
q_3 = \frac{M_{\bullet,3}}{M_{\bullet,1} + M_{\bullet,2}},
\]

(11.12)

then if \( q_3 \leq 2 \) then, if \( M_{\bullet,3} \leq M_{\bullet,2} \) we set

\[
a_3 = \frac{a_2}{1 + 0.4q_3},
\]

(11.13)

and define

\[
E_{\text{bind}} = \frac{GM_{\bullet,3}M_{\bullet,1}}{a_3},
\]

(11.14)

and

\[
\Delta K = 0.4q_3E_{\text{bind}},
\]

(11.15)

\( i = 3 \) and \( j = 2 \).

Otherwise if \( q_3 \leq 2 \) and \( M_{\bullet,3} > M_{\bullet,2} \) we set

\[
a_3 = \frac{a_3}{1 + 0.4q_3},
\]

(11.16)

and define

\[
E_{\text{bind}} = \frac{GM_{\bullet,3}M_{\bullet,1}}{a_2},
\]

(11.17)

and

\[
\Delta K = 0.4q_3E_{\text{bind}},
\]

(11.18)

\( i = 2 \) and \( j = 3 \).

Finally, if \( q_3 > 2 \), then we set

\[
a_3 = 0.53a_3,
\]

(11.19)

and define

\[
E_{\text{bind}} = \frac{GM_{\bullet,3}M_{\bullet,1}}{a_2},
\]

(11.20)

and

\[
\Delta K = 0.9q_3E_{\text{bind}},
\]

(11.21)

\( i = 2, j = 3 \).
11. Node Components

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 (11.22)

and

$$M_{\text{\textbullet, binary}} = M_{\text{\textbullet, j}} + M_{\text{\textbullet, 1}}.$$ \hfill (11.23)

The imparted velocities of these two systems are

$$V_{\text{\textbullet, ejected}} = \left[\frac{2\Delta K}{(1 + M_{\text{\textbullet, ejected}}/M_{\text{\textbullet, binary}})M_{\text{\textbullet, ejected}}}\right]^{1/2}$$ \hfill (11.24)

and

$$V_{\text{\textbullet, binary}} = \left[\frac{2\Delta K}{(1 + M_{\text{\textbullet, binary}}/M_{\text{\textbullet, ejected}})M_{\text{\textbullet, binary}}}\right]^{1/2}.$$ \hfill (11.25)

If

$$\frac{1}{2}V^{2}_{\text{\textbullet, ejected|binary}} + \Phi(d_{\text{\textbullet, ejected|binary}}) \geq 0$$ \hfill (11.26)

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 §12.39), 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 §16.4.1). In addition, the recoil velocity of the new black hole due to gravitational wave emission is computed using the selected method (see §12.41), 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 (11.27)

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 [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 blackHoleAccretionRate, blackHoleJetPower and blackHoleRadiativeEfficiency respectively.

If the [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.
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.

### 11.1.2. “Simple” Implementation

#### Properties

The simple black hole implementation defines the following property:

**mass** The mass of the black hole: $M_\bullet [\text{blackHoleMass}]$.

#### 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:

\[
\dot{M}_\bullet = (1 - \epsilon_{\text{wind}}) \epsilon_{\text{BH}} \dot{M}_{\bullet, \text{spheroid}}
\]

(11.28)

\[
\dot{M}_\bullet = \epsilon_{\text{BH}} \dot{M}_{\bullet, \text{spheroid}}
\]

(11.29)

where $\epsilon_{\text{BH}}$ 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 [blackHoleHeatsHotHalo]=true and [blackHoleAccretesFromHotHalo]=false then a power $\epsilon_{\text{heat}} \epsilon_{\text{BH}} \dot{M}_{\bullet, \text{spheroid}} c^2$ 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 $\epsilon_{\text{heat}} \epsilon_{\text{BH}} \dot{M}_{\bullet, \text{spheroid}}$. If [blackHoleHeatsHotHalo]=true and [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]

\[
L_{\text{wind}} = \epsilon_{\bullet, \text{wind}} \dot{M}_0 c^2,
\]

(11.30)

where $\epsilon_{\bullet, \text{wind}} = [\text{blackHoleWindEfficiency}]$ 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.
11. 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 §16.4.1).

Node promotion: None.

Additional Output

If the [blackHoleOutputAccretion] input parameter is set to true, then rest mass accretion rate (in $M_\odot$ Gyr$^{-1}$) is output as blackHoleAccretionRate.

11.2. Hot Halo

11.2.1. “Very Simple” Implementation

Properties

The very simple hot halo implementation defines the following properties:

- **mass**: The mass of gas in the hot halo: $M_{\text{hot}}$ [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:

$$M_{\text{hot}} = -\dot{M}_{\text{cooling}} + \dot{M}_{\text{outflow}},$$

(11.31)

where $\dot{M}_{\text{cooling}}$ is the rate of mass loss from the hot halo due to cooling (see §12.5.2). In the above $\dot{M}_{\text{outflow}}$ 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.
11.2. Hot Halo

11.2.2. “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 §12.1) and angular momentum based on the accreted mass and the halo spin parameter.
11. Node Components

**Differential Evolution**

In the standard hot halo implementation the hot gas mass and heavy element mass(es) evolves as:

\[
\begin{align*}
\dot{M}_{\text{failed}} &= \dot{M}_{\text{failed accretion}} \\
\dot{M}_{\text{hot}} &= \dot{M}_{\text{accretion}} - \dot{M}_{\text{cooling}} + \dot{M}_{\text{outflow, return}} - \dot{M}_{\text{expelled}} - \dot{M}_{\text{hot, stripped}}, \\
\dot{M}_{Z, \text{hot}} &= -\dot{M}_{\text{cooling}} \frac{M_{Z, \text{hot}}}{M_{\text{hot}}} + \dot{M}_{Z, \text{outflow, return}} - \dot{M}_{Z, \text{expelled}} - \dot{M}_{Z, \text{hot, stripped}}, \\
\dot{M}_{\text{chemical}} &= -\left[\dot{M}_{\text{cooling}} + \dot{M}_{\text{expelled}} + \dot{M}_{\text{hot, stripped}}\right] \frac{M_{\text{chemical}}}{M_{\text{hot}}} + f_{\text{chemical, outflow}} \dot{M}_{\text{outflow, return}} \\
\dot{M}_{\text{hot, stripped}} &= -4\pi \rho_{\text{hot}} (r_{\text{hot, outer}})^2 \dot{r}_{\text{hot, outer}} + \dot{M}_{\text{outflow}} f_{\text{outflow, stripped}}, \\
\dot{M}_{Z, \text{hot, stripped}} &= -4\pi \rho_{\text{hot}} (r_{\text{hot, outer}})^2 \dot{r}_{\text{hot, outer}} (M_{Z, \text{hot}}/M_{\text{hot}}) \\
&\quad + \dot{M}_{Z, \text{outflow}} f_{\text{outflow, stripped}},
\end{align*}
\]

where \(r_{\text{rp}}\) is the ram pressure stripping radius as computed by the `hotHaloRamPressureStrippingMethod` method (see §12.18), \(\dot{M}_{\text{accretion}}\) is the rate of growth of the hot component due to accretion from the intergalactic medium (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 §12.5.2—cooling rates are computed using the current node if `hotHaloCoolingFromNode` = `current node` or from the formation node if that parameter is set to `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 \([\text{hotHaloExcessHeatDrivesOutflow}]\) = `false` then the excess energy is not used and \(\dot{M}_{\text{expelled}} = 0\). Alternatively, if \([\text{hotHaloExcessHeatDrivesOutflow}]\) = `true` then

\[
\dot{M}_{\text{expelled}} = \begin{cases} 
\alpha_{\text{expel}} M_{\text{hot}}/\tau_{\text{dynamical}} & \text{if } \dot{M}_{\text{heating}} - \dot{M}_{\text{cool}} > \alpha_{\text{expel}} M_{\text{hot}}/\tau_{\text{dynamical}} \\
\dot{M}_{\text{heating}} - \dot{M}_{\text{cool}} & \text{otherwise},
\end{cases}
\]

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 \(\alpha_{\text{expel}} = [\text{hotHaloExpulsionRateMaximum}]\) 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 §12.22) 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 §12.28). The angular momentum of the hot gas evolves as:

\[
\dot{J}_{\text{hot}} = \dot{M}_{\text{accretion}} \frac{J_{\text{node}}}{M_{\text{node}}} - \dot{M}_{\text{cooling}} r_{\text{cool}} V_{\text{rotate}} + \dot{J}_{\text{outflow, return}} - \dot{M}_{\text{expelled}} \frac{J_{\text{hot}}}{M_{\text{hot}}},
\]

where \(\dot{M}_{\text{node}}\) and \(\dot{J}_{\text{node}}\) are defined in §11.5. For the outflowed components:

\[
\dot{M}_{\text{outflowed}} = -\dot{M}_{\text{outflow, return}} + \dot{M}_{\text{outflows}} (1 - f_{\text{outflow, stripped}}),
\]
\[ M_{\text{Z, outflowed}} = -M_{\text{Z, outflow return}} + M_{\text{Z, outflows}} (1 - f_{\text{outflow stripped}}), \]  

(11.43)  

and:

\[ J_{\text{outflowed}} = -J_{\text{outflow return}} + J_{\text{outflows}}. \]  

(11.44)  

In the above

\[ M | M_Z | J_{\text{outflow, return}} = \alpha_{\text{outflow return rate}} \frac{M | M_Z | J_{\text{outflowed}}}{\tau_{\text{dynamical, halo}}}, \]  

(11.45)  

\[ \text{where } \alpha_{\text{outflow return rate}} = (\text{hotHaloOutflowReturnRate}) \text{ is an input parameter controlling the rate at which gas flows from the outflowed to hot reservoirs, and } M | M_Z | J_{\text{outflows}} \text{ are the net rates of outflow from any components in the node.} \]

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

\[ f_{\text{outflow stripped}} = \epsilon_{\text{strip}} \frac{M_{\text{hot, outer}}}{M_{\text{hot, virial}}}, \]  

(11.47)  

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 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 \([\text{hotHaloTrackStrippedGas}] = \text{true}\)) then any stripped mass is transferred from the satellite galaxy to the hot halo of the host dark matter halo after each timestep. If \([\text{hotHaloNodeMergerLimitBaryonFraction}] = \text{true}\) 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 it 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 \([\text{hotHaloOutflowReturnOnFormation}] = \text{true}\) then all outflowed gas is returned to the hot gas reservoir on halo formation events (see §16.4.3).
11.3. Galactic Disk

11.3.1. “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:

\textbf{massGas} The mass of gas in the disk: \( M_{\text{disk, gas}} \)

\textbf{massStellar} The mass of stars in the disk: \( M_{\text{disk, stars}} \)

Initialization

No initialization is performed—disks are created as needed.

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 \( \tau_{\text{dyn}} \), where \( \tau_{\text{dyn}} \) is the dynamical time of the halo, and \( \tau_{\text{disk, star formation}, \text{minimum}} = [\text{diskStarFormationTimescaleMinimum}] \). 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 §12.37), but is limited to a maximum of \( M_{\text{disk, gas}}/\tau_{\text{disk, outflow, minimum}} \), where \( \tau_{\text{disk, outflow, minimum}} = [\text{diskOutflowTimescaleMinimum}] \). This outflow is piped to the hot halo component.

Event Evolution

\textit{Node mergers}: None

\textit{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 §12.14.1).

\textit{Node promotion}: None

11.3.2. “Exponential” Implementation

This implementation assumes a disk with an exponential surface density profile in which stars trace gas.
11.3. Galactic Disk

Properties

The exponential galactic disk implementation defines the following properties:

- **massGas** The mass of gas in the disk: \( M_{\text{disk, gas}} \) [diskMassGas].
- **abundancesGas** The mass of elements in the gaseous disk: \( M_{Z, \text{disk, gas}} \) [diskAbundancesGas{abundanceName}].
- **massStellar** The mass of stars in the disk: \( M_{\text{disk, stars}} \) [diskMassStellars].
- **abundancesStellar** The mass of elements in the stellar disk: \( M_{Z, \text{disk, stars}} \) [diskAbundancesStellar{abundanceName}].
- **luminositiesStellar** The luminosities (in multiple bands) of the stellar disk: \( L_{\text{disk, stars}} \) [diskLuminositiesStellar{luminosityName}].
- **angularMomentum** The angular momentum of the disk, \( J_{\text{disk}} \) [diskAngularMomentum].
- **radius** The radial scale length of the disk, \( R_{\text{disk}} \) [diskRadius].
- **velocity** The circular velocity of the disk at \( R_{\text{disk}} \), \( V_{\text{disk}} \) [diskVelocity].

Initialization

No initialization is performed—disks are created as needed.

Differential Evolution

In the exponential galactic disk implementation the gas mass evolves as:

\[
\dot{M}_{\text{disk, gas}} = \dot{M}_{\text{cooling}} - \dot{M}_{\text{outflow, disk}} - \dot{M}_{\text{stars, disk}} - \frac{\dot{M}_{\text{disk, gas}}}{\tau_{\text{bar}}},
\]

where the rate of change of stellar mass is

\[
\dot{M}_{\text{stars, disk}} = \Psi - \dot{R} - \frac{\dot{M}_{\text{stars, disk}}}{\tau_{\text{bar}}},
\]

with

\[
\Psi = \frac{M_{\text{disk, gas}}}{\tau_{\text{disk, star formation}}}
\]

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 §12.8). 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:

\[
\dot{M}_{Z, \text{disk, gas}} = \dot{M}_{Z, \text{cooling}} - \dot{M}_{Z, \text{outflow, disk}} - \dot{M}_{Z, \text{stars, disk}} + \dot{y},
\]

and

\[
\dot{M}_{Z, \text{stars, disk}} = \Psi \frac{M_{Z, \text{disk, gas}}}{M_{\text{disk, gas}}} - \dot{R}_{Z}
\]

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:

\[
\dot{J}_{\text{disk}} = \dot{J}_{\text{cooling}} - \left[ \dot{M}_{\text{outflow, disk}} + \frac{M_{\text{disk, gas}} + M_{\text{disk, stars}}}{\tau_{\text{bar}}} \right] \frac{J_{\text{disk}}}{M_{\text{disk, gas}} + M_{\text{disk, stars}}}. \]

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 §12.32) and prescriptions for expulsive and non-expulsive supernova feedback (see §12.38 and §12.37 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 $\alpha_{\text{outflow, minimum, disk}} = \frac{\text{diskOutflowTimescaleMinimum}}{}$ 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.

**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 §12.14.1).

*Node promotion:* None

**Additional Output**

If the [diskOutputStarFormationRate] input parameter is set to true, then the instantaneous star formation rate in the disk 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.:

$$
\frac{j(r)}{\langle j \rangle} = rV \frac{\int_0^\infty 2\pi r' \Sigma(r') r' V dr'}{\int_0^\infty 2\pi r' \Sigma(r') dr'}
$$

$$
\frac{j(r)}{\langle j \rangle} = r / 2r_{\text{disk}}.
$$

(11.57)

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:

$$
j(r) \rightarrow \left[ j^2(r) - \left( V_{\text{disk}}^2(r) r^2 - G M_{\text{disk}}(<r) r \right) \right]^{1/2},
$$

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.

11.4. Galactic Spheroid

**11.4.1. “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\(^4\).

\[^4\text{The spheroid density distribution is handled internally using a massDistribution object (see §16.5.4). As such, any mass distribution implemented as an extension of the massDistribution class (and which is described by a single length scale) is well supported.}\]
11.4. Galactic Spheroid

**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:

\[
\Sigma(R) \propto \exp \left( -b_n R^{1/n} \right),
\]

(11.59)

where the Sérsic index, \( n = [spheroidSersicIndex] \) 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.

**Properties**

The standard galactic spheroid implementation defines the following properties:

**masGas** The mass of gas in the spheroid: \( M_{spheroid, gas} [spheroidMassGas] \).

**abundancesGas** The mass of elements in the gaseous spheroid: \( M_{Z, spheroid, gas} [spheroidAbundancesGas{abundanceName}] \).

**massStellar** The mass of stars in the spheroid: \( M_{spheroid, stars} [spheroidMassStellar] \).

**abundancesStellar** The mass of elements in the stellar spheroid: \( M_{Z, spheroid, stars} [spheroidAbundancesStellar{abundanceName}] \).

**luminositiesStellar** The luminosities (in multiple bands) of the stellar spheroid: \( L_{spheroid, stars} [spheroidLuminositiesStellar{luminosityName}] \).

**angularMomentum** The pseudo-angular momentum\(^5\) of the spheroid, \( J_{spheroid} [spheroidAngularMomentum] \). The parameter \([spheroidAngularMomentumAtScaleRadius]\) controls the ratio of the specific pseudo-angular momentum at the scale radius of the standard spheroid to the mean specific pseudo-angular momentum. By default, this parameter is set to \([spheroidAngularMomentumAtScaleRadius] = I_2/I_3\), where

\[
I_n = \int_{0}^{\infty} \rho(r)r^n dr,
\]

(11.60)

and \( \rho(r) \) is the spheroid density profile, which is appropriate for a flat rotation curve. In some cases (e.g. the Hernquist profile) one or both of \( I_2 \) and \( I_3 \) can be infinite. In such cases \([spheroidAngularMomentumAtScaleRadius] = 0.5\) is assumed by default. If a finite truncation radius is assumed, or a different rotation curve is assumed, this ratio may be finite. The \([spheroidAngularMomentumAtScaleRadius]\) parameter allows control over these assumptions.

**radius** The radial scale length of the spheroid, \( r_{spheroid} [spheroidRadius] \).

**velocity** The circular velocity of the spheroid at \( r_{spheroid} \): \( V_{spheroid} [spheroidVelocity] \).

and the following pipes:

**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 \( M_\odot \) km\(^2\) s\(^{-2}\) 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).

\(^5\)Effectively the angular momentum that the spheroid would have, were it rotationally supported rather than pressure supported.
Initialization

No initialization is performed—spheroids are created as needed.

Differential Evolution

In the standard galactic spheroid implementation the gas mass evolves as:

\[
\dot{M}_{\text{spheroid, gas}} = -\dot{M}_{\text{outflow, spheroid}} - \dot{M}_{\text{stars, spheroid}},
\]

(11.61)

where the rate of change of stellar mass is

\[
\dot{M}_{\text{stars, spheroid}} = \Psi - \dot{R}
\]

(11.62)

with

\[
\Psi = \frac{M_{\text{spheroid, gas}}}{\tau_{\text{spheroid, star formation}}}
\]

(11.63)

with \(\tau_{\text{spheroid, star formation}}\) being the star formation timescale and \(\dot{R}\) is the rate of mass recycling from stars. Element abundances (including total metals) evolve according to:

\[
\dot{M}_{Z, \text{spheroid, gas}} = -\dot{M}_{Z, \text{outflow, spheroid}} - \dot{M}_{Z, \text{stars, spheroid}} + \dot{y},
\]

(11.64)

and

\[
\dot{M}_{Z, \text{stars, spheroid}} = \Psi \frac{M_{Z, \text{spheroid, gas}}}{M_{\text{spheroid, gas}}} - \dot{R}_Z
\]

(11.65)

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:

\[
\dot{J}_{\text{spheroid}} = \dot{M}_{\text{outflow, spheroid}} \frac{J_{\text{spheroid}}}{M_{\text{spheroid, gas}} + M_{\text{spheroid, stars}}}.
\]

(11.66)

The outflow rate, \(\dot{M}_{\text{outflow, spheroid}}\), is computed for the current star formation rate and gas properties by the stellar properties subsystem (see §12.3.2) and prescriptions for expulsive and non-expulsive supernova feedback (see §12.38 and §12.37 respectively), with an additional contribution given by

\[
\dot{M}_{\text{outflow, spheroid}} = \beta_{\text{spheroid, energy}} \frac{E_{\text{gas, spheroid}}}{V_{\text{spheroid}}^2}
\]

(11.67)

where \(\beta_{\text{spheroid, energy}} = \text{spheroidEnergeticOutflowMassRate}\) is an input parameter, and \(E_{\text{gas, spheroid}}\) is any input energy sent through the Tree_Node_Spheroid_Gas_Energy_Input pipe, but is not allowed to exceed \(M_{\text{gas, spheroid}}/\tau_{\text{outflow minimum, spheroid}}\tau_{\text{spheroid, dynamical}}\), where \(\tau_{\text{spheroid, dynamical}} = R_{\text{spheroid}}/V_{\text{spheroid}}\) is the dynamical time of the spheroid and \(\tau_{\text{outflow minimum, spheroid}} = \text{spheroidOutflowTimescaleMinimum}\) 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.

\(^{6}\)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 §11.3.2). 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.
11.5. Basic Properties

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 §16.4.1).

Node promotion: None.

Additional Output

If the [spheroidOutputStarFormationRate] input parameter is set to true, then the instantaneous star formation rate in the spheroid will be included in the output, as spheroidStarFormationRate.

11.5. Basic Properties

Basic properties are the total mass of a node and the cosmic time at which it currently exists.

11.5.1. “Non-evolving” Implementation

Properties

The non-evolving basic properties implementation defines the following properties:

- mass The total mass of the node: $M_{\text{node}}$ [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): [basicTimeLastIsolated].

Initialization

All basic properties are required to be initialized by the merger tree construction routine.

Differential Evolution

Properties are evolved according to:

$$\dot{M}_{\text{node}} = 0$$  \hspace{1cm} (11.68)
$$\dot{t}_{\text{node}} = 1.$$  \hspace{1cm} (11.69)

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.
11. Node Components

11.5.2. “Simple” Implementation

Properties

The simple basic properties implementation defines the following properties:

mass The total mass of the node: $M_{\text{node}}$ [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): [basicTimeLastIsolated].

Initialization

All basic properties are required to be initialized by the merger tree construction routine.

Differential Evolution

Properties are evolved according to:

$$M_{\text{node}}' = \begin{cases} \frac{M_{\text{node,parent}} - M_{\text{node}}}{t_{\text{node,parent}} - t_{\text{node}}} & \text{if primary progenitor} \\ 0 & \text{otherwise,} \end{cases}$$

(11.70)

$$t_{\text{node}}' = 1,$$

(11.71)

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.

11.6. Position

The position component implements the position and velocity of each galaxy. See §10.2 for important notes on velocity definitions in GALACTICUS.

11.6.1. “Preset” Implementation

Properties

The preset position implementation defines the following properties:

position The 3-D position of the node: $x$ [positionPosition[X|Y|Z]].

velocity The 3-D velocity of the node: $v$ [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\(^7\).

Event Evolution
Node mergers: None.
Satellite merging: None.
Node promotion: The position and velocity are updated to those of the parent node.

11.7. Satellite Orbit
This component tracks the orbital properties of subhalos.

11.7.1. “Preset” Implementation
Properties
The preset satellite orbit implementation defines the following properties:

- **mergTime** The time until the satellite will merge with its host: \( t_\text{satellite,merge} \) [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 boundMassHistory property).
- **boundMassHistory** A history time-series of the total bound mass of the satellite.
- **virialOrbit** The orbit (a keplerOrbit object; see §16.5.1) 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.

---

\(^7\)While 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.
11. Node Components

**Event Evolution**

*Node mergers:* None.

*Satellite merging:* None.

*Node promotion:* None.

### 11.7.2. "Very Simple" Implementation

**Properties**
The simple satellite orbit implementation defines the following properties:

- **mergeTime** The time until the satellite will merge with its host: $t_{\text{satellite,merge}}$ [satelliteMergeTime].

**Initialization**
None.

**Differential Evolution**
Properties are evolved according to:

$$
\dot{t}_{\text{satellite,merge}} = -1.
$$

(11.72)

**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).

### 11.7.3. "Simple" Implementation

**Properties**
The simple satellite orbit implementation defines the following properties:

- **mergeTime** The time until the satellite will merge with its host: $t_{\text{satellite,merge}}$ [satelliteMergeTime].
- **boundMass** The remaining, total bound mass of the satellite: $M_{\text{node,bind}}$ [satelliteBoundMass].
- **virialOrbit** The orbit (returned as a keplerOrbit object; see §16.5.1) of the satellite at the point of virial radius crossing.

**Initialization**
None.
11.8. Dark Matter Halo Spin

Differential Evolution

Properties are evolved according to:

\[ t_{\text{satellite,merge}} = -1, \]

with \( \dot{M}_{\text{node,bound}} \) set to the rate given by the `darkMatterHaloMassLossRateMethod` method (see §16.4.1). 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 §16.4.1) 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 §11.11).

Satellite merging: None.

Node promotion: Not applicable (component only exists for satellite nodes).

11.8. Dark Matter Halo Spin

11.8.1. “Random” Implementation

Properties

The random dark matter halo spin implementation defines the following properties:

spin The spin parameter of the halo: \( \lambda [\text{spinSpin}] \).

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 `[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
11. Node Components

selected that a new spin value was chosen.)

11.8.2. “Preset” Implementation

Properties
The preset dark matter halo spin implementation defines the following properties:

spin The spin parameter of the halo: \( \lambda \) [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.

11.9. Dark Matter Profile

This component stores dynamic properties associated with dark matter halo density profiles.

11.9.1. “Scale” Implementation

Properties
The scale dark matter profile implementation defines the following properties:

scale The scale length of the density profile \([\text{darkMatterProfileScale}]\);

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 §12.7.3), but is not allowed to drop below \([\text{darkMatterProfileMinimumConcentration}]\), such that the scale length is equal to the virial radius divided by that concentration.

Differential Evolution
The scale radius does not evolve.
### 11.9.2. “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 §12.7.3), 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 §16.4.1).

#### Differential Evolution

The scale radius and shape parameters do not evolve.

### 11.10. Merging Statistics

This component records statistics associated with galaxy merging.

#### 11.10.1. “Standard Implementation

#### Properties

The standard merging statistics implementation defines the following properties:
11. Node Components

**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.

**11.10.2. “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 $\Delta t = [nodeRecentMajorMergerInterval]_{\text{abs}}$ if [nodeRecentMajorMergerIntervalType]=absolute, or $\Delta t = [nodeRecentMajorMergerInterval]_{\tau_{\text{dyn}}}$ (where $\tau_{\text{dyn}}$ is the halo dynamical time) if [nodeRecentMajorMergerIntervalType]=dynamical. [mergingStatisticsRecentMajorMergerTime];

**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.

11.11. Formation Times

This component implements “formation times” of dark matter halos.

11.11.1. “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.

11.12. Indices

This component tracks various indices associated with nodes.
11. Node Components

11.12.1. “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.

11.13. Inter-Output Quantities

This component tracks various quantities averaged between successive outputs.

11.13.1. “Standard” Implementation

Properties

The ‘standard’ inter-output implementation defines the following properties:

diskStarFormationRate The mean star formation rate in the disk between the previous and current output times.

spheroidStarFormationRate The mean star formation rate 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:

\[
\frac{d}{dt} \langle \dot{M}_* \rangle = \frac{\dot{M}_*}{\Delta t},
\]

(11.74)

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.
12. Physical Implementations

12.1. 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 \texttt{accretionHalosMethod} input parameter.

12.1.1. Simple Method

Selected using \texttt{accretionHalosMethod=simple}, this method sets the accretion rate of baryons into a halo to be:

\[
\dot{M}_{\text{accretion}} = \begin{cases} 
(\Omega_b/\Omega_M)\dot{M}_{\text{halo}} & \text{if } V_{\text{virial}} > V_{\text{reionization}} \text{ or } z > z_{\text{reionization}} \\
0 & \text{otherwise}
\end{cases}
\]  

(12.1)

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 §12.22).

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 \texttt{[accretionHalosSimpleNegativeAccretionAllowed]=false}.

12.1.2. Null Method

Selected using \texttt{accretionHalosMethod=null}, this method sets the accretion rate of baryons into a halo to be zero always.

12.2. Background Cosmology

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 \texttt{cosmologyMethod}, and the physics that must be implemented for each cosmological model is described in more detail in §16.4.1. Currently implemented cosmological models are as follows.
12. Physical Implementations

12.2.1. 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.

12.2.2. 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 $w_1 = [\text{darkEnergyEquationOfStateW1}]$.

12.3. 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 §16.4.1. Current implementations of accretion disks are as follows.

12.3.1. 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).

12.3.2. 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 $j \rightarrow 1$, 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:

$$\alpha(j) = 0.015 + 0.02j^4 \quad \text{if} \quad g = \exp(\omega \tau) \text{ and } E = E_{\text{ISCO}},$$  
(12.2)

$$\alpha(j) = 0.025 + 0.08j^4 \quad \text{if} \quad g = 1 + \omega \tau \text{ and } E = E_{\text{ISCO}},$$  
(12.3)

$$\alpha(j) = 0.010 + 0.00j^4 \quad \text{if} \quad g = \exp(\omega \tau) \text{ and } E = 1,$$

(12.4)

$$\alpha(j) = 0.025 + 0.02j^4 \quad \text{if} \quad g = 1 + \omega \tau \text{ and } E = 1.$$  
(12.5)

### 12.3.3. “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:

$$f_{\text{ADAF}} = \left[1 + \exp(y_{\text{min}})\right]^{-1} + \left[1 + \exp(y_{\text{max}})\right]^{-1},$$  
(12.6)

where

$$y_{\text{min}} = \log(x/x_{\text{min}})/\Delta x,$$  
(12.7)

$$y_{\text{max}} = -\log(x/x_{\text{max}})/\Delta x,$$

(12.8)

and,

$$x = \dot{M}/\dot{M}_{\text{Eddington}}.$$  
(12.9)

Here, $x_{\text{min}} = [\text{accretionRateThinDiskMinimum}]$, $x_{\text{max}} = [\text{accretionRateThinDiskMaximum}]$, and $\Delta x = [\text{accretionRateTransitionWidth}]$. If either $[\text{accretionRateThinDiskMinimum}]$ or $[\text{accretionRateThinDiskMaximum}]$ is set to “none” then the corresponding term in eqn. (12.6) 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 $x < x_{\text{min}}$.

### 12.3.4. 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 [accretionDiskJetPowerEddington] of the Eddington luminosity. The radiative efficiency is similarly set at a fixed value of [accretionDiskRadiativeEfficiencyEddington]. Since no physical model for the disk is assumed, the black hole spin up rate is always set to zero.

### 12.4. Dark Matter Structure Formation

A variety of functions are used to describe structure formation in dark matter dominated universes. These are described below.

#### 12.4.1. 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 §12.4.5) and normalized to a value of $\sigma_8$ specified by [sigma_8].
12. Physical Implementations

(Running) Power Law Spectrum

Selected via powerSpectrumMethod=powerLaw, this method implements a primordial power spectrum of the form:

\[ P(k) \propto k^{n_{\text{eff}}(k)}, \]  

(12.10)

where

\[ n_{\text{eff}}(k) = n_s + \frac{1}{2} \frac{\text{d} n}{\text{d} \ln k} \ln \left( \frac{k}{k_{\text{ref}}} \right), \]  

(12.11)

where \( n_s \) = powerSpectrumIndex is the power spectrum index at wavenumber \( k_{\text{ref}} \) = powerSpectrumReferenceWavenumber and \( \frac{\text{d} n}{\text{d} \ln k} \) = powerSpectrumRunning describes the running of this index with wavenumber.

12.4.2. 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:

\[ \sigma^2(M) = \frac{1}{2\pi^2} \int_0^{\infty} P(k)T^2(k)W^2(k)k^2 \text{d}k \]  

(12.12)

where \( P(k) \) is the primordial power spectrum (see §12.4.1), \( T(k) \) is the transfer function (see §12.4.5), and \( W(k) \) is the power spectrum variance window function (see §12.4.3).

12.4.3. 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:

\[ W(k) = \frac{3(\sin(x) - x \cos(x))}{x^3}, \]  

(12.13)

where \( x = kR \) and \( R = (3M/4\pi\bar{\rho})^{1/3} \) for a smoothing scale \( M \) and mean matter density \( \bar{\rho} \).

Sharp in \( k \)-space

Selected via powerSpectrumWindowFunctionMethod=kSpaceSharp, this method implements a top-hat window function in \( k \)-space:

\[ W(k) = \begin{cases} 
1 & \text{if } k < k_s, \\
0 & \text{if } k > k_s,
\end{cases} \]  

(12.14)

where if [powerSpectrumWindowFunctionSharpKSpaceNormalization]=natural then \( k_s = (6\Pi^2\bar{\rho}/M)^{1/3} \) for a smoothing scale \( M \) and mean matter density \( \bar{\rho} \). Otherwise, [powerSpectrumWindowFunctionSharpKSpaceNormalization] must be set to a numerical value, \( \alpha \), in which case \( k_s = \alpha/R_{\text{th}} \) with \( R_{\text{th}} = 3M/4\pi\bar{\rho} \) for a smoothing scale \( M \) and mean matter density \( \bar{\rho} \).
12.4. Dark Matter Structure Formation

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:

\[
W(k) = W_{th}(k)W_s(k),
\]

where

\[
W(k) = \frac{3(\sin(x) - x \cos(x))}{x^3},
\]

where \(x = kR_{th}\), and

\[
W_s(k) = \begin{cases} 
1 & \text{if } k < k_s \\
0 & \text{if } k > k_s,
\end{cases}
\]

where \(k_s = \alpha/R_s\) if `powerSpectrumWindowFunctionSharpKSpaceNormalization` is assigned a numerical value. Alternatively, if `powerSpectrumWindowFunctionSharpKSpaceNormalization=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:

\[
R^2_{th} + R^2_s = (3M/4\pi\bar{\rho})^{2/3}
\]

\[
R_s = \beta R_{th},
\]

where \(\beta = [powerSpectrumWindowFunctionSharpKSpaceTopHatRadiiRatio]\).

12.4.4. 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.

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.

12.4.5. 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 §12.4.1) 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 \(k\).
12. Physical Implementations

**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]:

\[
T(k) \rightarrow T(k)(1 + [kR_0^2\nu])^{-\eta/\nu},
\]

where \(R_0^2 = [transferFunctionWdmCutOffScale], \epsilon = [transferFunctionWdmEpsilon], \eta = [transferFunctionWdmEta], \nu = [transferFunctionWdmNu].

**CMBFast**

Selected with `transferFunctionMethod=CMBFast`, this method uses the CMBFast 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. CMBFast 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 §16.4.1.

12.4.6. 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.

12.4.7. 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 \texttt{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 \texttt{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,

\[
\delta_{\text{crit}}(t) = \frac{3(12\pi)^{2/3}}{20} \left[ 1 + 0.0123 \log_{10} \left\{ \Omega_{\text{matter}}(t) \right\} / D(t) \right].
\] (12.21)

12.4.8. 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 \texttt{criticalOverdensityMassScalingMethod} parameter.

Null

Selected with \texttt{criticalOverdensityMassScalingMethod=null} this method assumes that the critical overdensity is independent of mass.

Warm Dark Matter

Selected with \texttt{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

\[
\exp \left[ \left( \frac{M_J}{8M} \right)^{1.40} + \left( \frac{M_J}{8M} \right)^{0.45} \right],
\] (12.22)

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]:

\[
M_J = 3.06 \times 10^8 \left( \frac{1 + z_{eq}}{3000} \right)^{1.5} \left( \frac{\Omega_M b_0^2}{0.15} \right)^{1/2} \left( \frac{g_X}{1.5} \right)^{-1} (m_X/1.0\text{keV})^{-4},
\] (12.23)

the redshift of matter-radiation equality is given by

\[
z_{eq} = 3600 \left( \frac{\Omega_M b_0^2}{0.15} \right) - 1,
\] (12.24)

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.

12.4.9. Virial Density Contrast

The method used to compute the mean density contrast of virialized dark matter halos is selected via the \texttt{virialDensityContrastMethod} parameter.
12. Physical Implementations

**Bryan & Norman (1998) Fitting Function**

Selected with \texttt{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 \texttt{virialDensityContrastMethod=fixed} this method uses a fixed virial density contrast of \texttt{[virialDensityContrastFixed]}, defined relative to \texttt{criticalDensity} and \texttt{meanDensity} as specified by \texttt{[virialDensityContrastFixedType]}.

**Spherical Collapse (Matter + Cosmological Constant)**

Selected with \texttt{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 \texttt{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,

\[
\Delta_{\text{virial}}(t) = 18\pi^2 \left[ 1 + 0.4093 \left\{ \frac{1}{\Omega_{\text{matter}}(t)} - 1 \right\} (t^{0.9052}). \right. \tag{12.25}
\]

12.4.10. Halo Bias

The dark matter halo linear bias method is selected via the \texttt{darkMatterHaloBiasMethod} parameter.

**Press-Schechter**

Selected with \texttt{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 \texttt{darkMatterHaloBiasMethod=SMT} this method uses a bias consistent with the halo mass function of Sheth et al. [2001].

**Tinker**

Selected with \texttt{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 §12.4.9).

12.4.11. 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 \texttt{haloMassFunctionMethod} parameter.
12.5. Cooling of Gas Inside Halos

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,

\[
n(M, t) = 2\Omega_\Lambda^2 \rho_{\text{crit}} M^α \sigma^2(M) f(S(M, t)),
\]

(12.26)

where \( \alpha = \frac{d \ln \sigma}{d \ln M} \) and \( f[S] \) is the excursion set barrier first crossing distribution for variance \( S(M) = \sigma^2(M) \), computed using the selected `excursionSetFirstCrossingMethod` (see §16.4.1).

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 §12.4.9).

12.5. 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.

12.5.1. 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 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 §16.4.1. The cooling function is assumed to be computed under conditions of collisional ionization equilibrium and therefore to scale as \( \rho^2 \).
Table 12.1.: Coefficients of $H_2^+$ cooling functions as appearing in the fitting function, eq. 12.28.

<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>

CMB Compton Cooling

Selected using `coolingFunctionMethods=CMB_Compton`, this method computes the cooling function due to Compton scattering off of CMB photons:

$$\Lambda = \frac{4\sigma_Tn_eK_B}{m_e c}T_{\text{CMB}}^4(T - T_{\text{CMB}}),$$  \hspace{1cm} (12.27)

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, $T_{\text{CMB}}$ 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 §12.22).

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 H–H$_2$ cooling function, the fitting functions from Galli and Palla [1998] are used. For the $H_2^+ - e^-$ and H–H$_2^+$ cooling functions fitting functions to the results plotted in Suchkov and Shechekinov [1978] are used:

$$\log_{10} \left( \frac{\Lambda(T)}{\text{erg s}^{-1} \text{cm}^3} \right) = C_0 + C_1 \log_{10} \left( \frac{T}{K} \right) + C_2 \left[ \log_{10} \left( \frac{T}{K} \right) \right]^2,$$  \hspace{1cm} (12.28)

where the coefficients $C_{0-2}$ are given in Table 12.1.

12.5.2. 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

$$\dot{M}_{\text{cool}} = \begin{cases} 
4\pi r_{\text{infall}}^2 \rho(r_{\text{infall}}) v_{\text{infall}} & \text{if } r_{\text{infall}} < r_{\text{hot,outer}} \\
0 & \text{if } r_{\text{infall}} \geq r_{\text{hot,outer}},
\end{cases}$$  \hspace{1cm} (12.29)

where $\rho(r)$ is the density profile of the hot halo, and $r_{\text{infall}}$ is the infall radius (see §12.5.6). The cooling rate is also set to zero in halos with virial velocities below $[\text{coolingCutOffVelocity}]$ at redshifts below $[\text{coolingCutOffRedshift}]$. 

188
Simple

Selected with `coolingRateMethod=simple`, this method computes the cooling rate using

\[ \dot{M}_{\text{cool}} = M_{\text{hot}} / \tau_{\text{cool}}, \]  

(12.30)

where \( \tau_{\text{cool}} = \text{[coolingRateSimpleTimescale]} \).

Simple Scaling

Selected with `coolingRateMethod=simpleScaling`, this method computes the cooling rate using

\[ \dot{M}_{\text{cool}} = M_{\text{hot}} / \tau_{\text{cool}}(M_{\text{halo}}, z), \]  

(12.31)

where

\[ \tau_{\text{cool}} = \tau_{\text{cool},0}(1 + z)^{\beta_{\text{cool}}} \exp \left( \frac{M_{\text{halo}}}{M_{\text{transition}}} \right), \]  

(12.32)

\( \tau_{\text{cool},0} = \text{[coolingRateSimpleScalingTimescale]}, \beta_{\text{cool}} = \text{[coolingRateSimpleScalingTimescaleExponent]}, \) and \( M_{\text{transition}} = \text{[coolingRateSimpleScalingTransitionMass]} \).

White & Frenk

Selected with `coolingRateMethod=White-Frenk1991`, this method computes the cooling rate using the expression given by White and Frenk [1991], namely

\[ \dot{M}_{\text{cool}} = \begin{cases} 4\pi r_{\text{infall}}^2 \rho(r_{\text{infall}}) r_{\text{infall}} & \text{if } r_{\text{infall}} < r_{\text{hot,outer}} \\ M_{\text{hot}} / (r_{\text{halo,dynamical}}) & \text{if } r_{\text{infall}} \geq r_{\text{hot,outer}}, \end{cases} \]  

(12.33)

where \( r_{\text{infall}} \) is the infall radius (see §12.5.6) in the hot halo and \( \rho(r) \) is the density profile of the hot halo.

12.5.3. 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 \( \text{[coolingCutOffVelocity]} \) at redshifts below/above \( \text{[coolingCutOffRedshift]} \) for \( \text{[coolingCutOffWhen]}=\text{after}/\text{before} \). In other halos the cooling rate is not modified.

12.5.4. 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 §12.5.9) equals the cooling time (see §12.5.8). 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.
12. Physical Implementations

**Isothermal**

Selected with \texttt{coolingRadiusMethod=isonothermal}, 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:

\[ t_{cool} \equiv \frac{E}{E} \propto \rho(r)^{-1}. \quad (12.34) \]

The cooling radius is then derived using

\[ \rho(r_{cool}) \propto t_{available}^{-1} \quad (12.35) \]

which implies

\[ r_{cool} = r_{virial} \left( \frac{t_{available}}{t_{cool, virial}} \right)^{1/2}, \quad (12.36) \]

where \( t_{cool, virial} \) is the cooling time at the virial radius.

12.5.5. 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 §12.5.10).

12.5.6. 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 §12.5.4).

**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 §12.5.4 and §12.5.5).

**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 §12.5.10).

12.5.7. Cooling Specific Angular Momentum

The algorithm used to compute the specific angular momentum of cooling gas is selected via the \texttt{coolingSpecificAngularMomentumMethod} parameter.
12.5. Cooling of Gas Inside Halos

Constant Rotation

Selected with `coolingSpecificAngularMomentumMethod=constantRotation`, this implementation assumes that the specific angular momentum of cooling gas is given either by

\[ j_{\text{cool}} = \langle j \rangle r_{\text{cool}} A, \]  

(12.37)

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 \([coolingAngularMomentumUseInteriorMean]=false\), or by

\[ j_{\text{cool}} = \langle j \rangle I_3(r_{\text{cool}})/I_2(r_{\text{cool}}) A, \]  

(12.38)

where \( I_n(r) \) is the \( n^{\text{th}} \) 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 \([coolingAngularMomentumUseInteriorMean]=true\).

If \([coolingMeanAngularMomentumFrom]=darkMatter\) then \( \langle j \rangle \) is the mean specific angular momentum of the dark matter halo, computed from its spin parameter, while if \([coolingMeanAngularMomentumFrom]=hotGas\) then \( \langle j \rangle \) is equal to the mean specific angular momentum of gas currently in the hot gas reservoir. If \([coolingRotationVelocityFrom]=darkMatter\) then the rotation normalization \( A \) is computed using the dark matter density profile, while if \([coolingRotationVelocityFrom]=hotGas\) it is computed using the density profile of the hot gas reservoir.

Mean

Selected with `coolingSpecificAngularMomentumMethod=mean`, this assumes that the specific angular momentum of cooling gas is given by

\[ j_{\text{cool}} = J_{\text{hot}}/M_{\text{hot}}, \]  

(12.39)

where \( J_{\text{hot}} \) and \( M_{\text{hot}} \) are the total angular momentum and mass of the hot halo respectively.

12.5.8. Cooling Time

The algorithm used to compute the time taken for gas to cool (i.e. the cooling time) is selected via the `coolingTimeMethod` parameter.

Simple

Selected with `coolingTimeMethod=simple`, this method assumes that the cooling time is simply

\[ t_{\text{cool}} = \frac{N k_B T n_{\text{tot}}}{2 \Lambda}, \]  

(12.40)

where \( N =coolingTimeSimpleDegreesOfFreedom \) is the number of degrees of freedom in the cooling gas which has temperature \( T \) and total particle number density (including electrons) \( n_{\text{tot}} \) and \( \Lambda \) is the cooling function.

12.5.9. 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 `coolingTimeAvailableMethod` parameter.

Halo Formation

Selected with `coolingTimeAvailableMethod=haloFormation`, this method assumes that the time available for cooling is equal to

\[ t_{\text{available}} = t - t_{\text{form}}, \]  

(12.41)

where \( t_{\text{form}} \) is the time at which the halo formed (see §11.11).
White & Frenk (1991)

Selected with \texttt{coolingTimeAvailableMethod=White-Frenk1991}, this method assumes that the time available for cooling is equal to

\[ t_{\text{available}} = \exp\left[ f \ln t_{\text{Universe}} + (1 - f) \ln t_{\text{dynamical}} \right], \tag{12.42} \]

where \( f = \text{coolingTimeAvailableAgeFactor} \) is an interpolating factor, \( t_{\text{Universe}} \) is the age of the Universe and \( t_{\text{dynamical}} \) is the dynamical time in the halo. The original White and Frenk [1991] algorithm corresponds to \( f = 1 \).

12.5.10. 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 \texttt{freefallTimeAvailableMethod} parameter.

Halo Formation

Selected with \texttt{freefallTimeAvailableMethod=haloFormation}, this method assumes that the time available for freefall is equal to

\[ t_{\text{available}} = t - t_{\text{form}}, \tag{12.43} \]

where \( t_{\text{form}} \) is the time at which the halo formed (see §11.11).

12.6. Cosmology

The method used to compute cosmological relations (e.g. expansion factor as a function of time) is selected by the \texttt{cosmologyMethod} parameter.

12.6.1. Matter + Cosmological Constant Universes

Selected with \texttt{cosmologyMethod=matter-lambda}, this method assumes a universe which contains only matter and a cosmological constant.

12.7. Dark Matter Halos

Several algorithms are used to implement dark matter halos.

12.7.1. Mass Accretion History

The method used to compute mass accretion histories of dark matter halos is selected via the \texttt{darkMatterAccretionHistoryMethod} parameter.

Wechsler et al. (2002)

Selected with \texttt{darkMatterAccretionHistoryMethod=Wechsler2002}, under this method the mass accretion history is given by [Wechsler et al., 2002]:

\[ M(t) = M(t_0) \exp \left( -2a \left[ \frac{a(t_0)}{a(t)} - 1 \right] \right), \tag{12.44} \]
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

\[
\frac{dM}{dt} = \left( \frac{d \ln \sigma(M)}{d \ln M} \right)^{-1} \left( \frac{d \delta_c(t)}{dt} \right) \left( \frac{M}{\delta_c(t)} \right) \left( \frac{d \ln \sigma(M)}{d \ln \delta_c(t)} \right).
\] (12.45)

This differential equation is then solved numerically to find the mass accretion history.

12.7.2. Density Profile

The method uses to compute density profiles of dark matter halos is selected via the darkMatterProfileMethod parameter.

Isothermal

Selected with darkMatterProfileMethod=isothermal, under this method the density profile is given by:

\[
\rho_{\text{darkmatter}}(r) \propto r^{-2},
\] (12.46)

normalized such that the total mass of the node is enclosed with the virial radius.

NFW

Selected with darkMatterProfileMethod=NFW, under this method the Navarro-Frenk-White (dark matter halo profile) (NFW) density profile [Navarro et al., 1997] is used

\[
\rho_{\text{darkmatter}}(r) \propto \left( \frac{r}{r_s} \right)^{-1} \left[ 1 + \left( \frac{r}{r_s} \right) \right]^{-2},
\] (12.47)

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 §12.7.3).

Einasto

Selected with darkMatterProfileMethod=Einasto, under this method the Einasto density profile (e.g. Cardone et al. 2005) is used

\[
\rho_{\text{darkmatter}}(r) = \rho_{-2} \exp \left( -\frac{2}{\alpha} \left[ \left( \frac{r}{r_{-2}} \right)^\alpha - 1 \right] \right),
\] (12.48)

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 §12.7.3). The shape parameter, \( \alpha \), is set using the density profile shape method (see §12.7.4).
12.7.3. 

**Density Profile Concentration**

The method uses to compute the concentrations of dark matter profiles is selected via the `darkMatterConcentrationMethod` parameter.

**Navarro, Frenk & White (1996)**

Selected with `darkMatterConcentrationMethod=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

\[
\delta_{\text{crit}}(t_{\text{form}}) = \left[ 2\nu_{1/2}^2 \left\{ \sigma(fM)^2 - \sigma(M)^2 \right\} \right]^{1/2} + \delta_{\text{crit}}(t_0),
\]

where \( \nu_{1/2} = [\text{erfc}^{-1}(1/2)]^{1/2} \). Navarro et al. [1996] then assume an overdensity at collapse of

\[
\Delta(t_{\text{form}}) = C \left[ \frac{a(t_0)}{a(t_{\text{form}})} \right]^3
\]

where \( C = \text{[nfw96ConcentrationC]} \) is a parameter of the algorithm. The concentration is then determined by solving

\[
\frac{\Delta(t_{\text{form}})}{\Delta_{\text{virial}}(t_0)} = \frac{c^3}{3[\ln(1 + c) - c/(1 + c)]}.
\]

**Gao (2008)**

Selected with `darkMatterConcentrationMethod=Gao2008`, under this method the concentration is computed using a fitting function from Gao et al. [2008]:

\[
\log_{10} c = A \log_{10} M_{\text{halo}} + B.
\]

The parameters are a function of expansion factor, \( a \). We use the following fits to the Gao et al. [2008] results:

\[
A = -0.140 \exp \left[ - \left( \frac{\log_{10} a + 0.05}{0.35} \right)^2 \right],
\]

\[
B = 2.646 \exp \left[ - \left( \frac{\log_{10} a/0.50}{2} \right)^2 \right].
\]

**Zhao (2009)**

Selected with `darkMatterConcentrationMethod=Zhao2009`, under this method the concentration is computed using a fitting function from Zhao et al. [2009]:

\[
c = 4 \left( 1 + \left[ \frac{t}{3.75t_{\text{form}}} \right]^{8.4} \right)^{1/8},
\]

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 §16.4.1).
12.7. Dark Matter Halos

Muñoz-Cuartas (2011)

Selected with `darkMatterConcentrationMethod=Munoz-Cuartas2011`, under this method the concentration is computed using a fitting function from Muñoz-Cuartas et al. [2011]:

$$\log_{10} c = a \log_{10} \left( \frac{M_{\text{halo}}}{h^{-1} M_\odot} \right) + b.$$  \hfill (12.56)

The parameters are a function of redshift, $z$, given by

$$a = wz - m,$$  \hfill (12.57)

$$b = \frac{\alpha}{(z + \gamma)} + \frac{\beta}{(z + \gamma)^2},$$  \hfill (12.58)

where $w = 0.029$, $m = 0.097$, $\alpha = -110.001$, $\beta = 2469.720$, $\gamma = 16.885$.

Prada et al. (2011)

Selected with `darkMatterConcentrationMethod=Prada2011`, under this method the concentration is computed using a fitting function from Prada et al. [2011]:

$$c(M,t) = B_0(x)C(\sigma'),$$  \hfill (12.59)

where

$$\sigma'(M,t) = B_1(x)\sigma(M,t),$$  \hfill (12.60)

$$B_0(x) = c_{\text{min}}(x)/c_{\text{min}}(1.393),$$  \hfill (12.61)

$$B_1(x) = \sigma_{\text{min}}^{-1}(x)/\sigma_{\text{min}}^{-1}(1.393),$$  \hfill (12.62)

$$c_{\text{min}}(x) = c_0 + (c_1 - c_0)[\tan^{-1}\{\alpha(x - x_0)\}/\pi + 1/2],$$  \hfill (12.63)

$$\sigma_{\text{min}}^{-1}(x) = \sigma_0^{-1} + (\sigma_1^{-1} - \sigma_0^{-1})[\tan^{-1}\{\beta(x - x_1)\}/\pi + 1/2],$$  \hfill (12.64)

$$C(\sigma') = A(\sigma'/b)^c + 1\exp(d/\sigma'^2),$$  \hfill (12.65)

$$x = (\Omega_\Lambda/\Omega_M)^{1/3}a(t),$$  \hfill (12.66)

with the following parameters (default values taken from Prada et al. [2011] given in []): $A = \text{[prada2011ConcentrationA]} = 2.881$, $b = \text{[prada2011ConcentrationB]} = 1.257$, $c = \text{[prada2011ConcentrationC]} = 1.022$, $d = \text{[prada2011ConcentrationD]} = 0.060$, $c_0 = \text{[prada2011ConcentrationC0]} = 3.681$, $c_1 = \text{[prada2011ConcentrationC1]} = 5.033$, $x_0 = \text{[prada2011ConcentrationX0]} = 0.424$, $x_1 = \text{[prada2011ConcentrationX1]} = 0.526$, $\sigma_0^{-1} = \text{[prada2011ConcentrationInverseSigma0]} = 1.047$, $\sigma_1^{-1} = \text{[prada2011ConcentrationInverseSigma1]} = 1.646$, $\alpha = \text{[prada2011ConcentrationAlpha]} = 6.948$, and $\beta = \text{[prada2011ConcentrationBeta]} = 7.386$.

12.7.4. 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]:

$$\alpha = \begin{cases} 
0.155 + 0.0095\nu^2 & \text{if } \nu < 3.907 \\
0.3 & \text{if } \nu \geq 3.907
\end{cases}$$  \hfill (12.67)

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.
12. Physical Implementations

12.7.5. 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:

\[
\dot{M}_{\text{node, bound}} = -\frac{M_{\text{node, bound}}}{\tau} \left( \frac{M_{\text{node, bound}}}{M_{\text{node, parent}}} \right)^\zeta, \tag{12.68}
\]

where \(M_{\text{node, parent}}\) is the mass of the parent node in which the halo lives and

\[
\tau = \tau_0 \left( \frac{\Delta_{\text{vir}}(t)}{\Delta(t_0)} \right)^{-1/2} a^{3/2}, \tag{12.69}
\]

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 \((\rho_{\text{vir}})^{-1/2}\) where \(\langle \rho_{\text{vir}} \rangle\) is the mean virial overdensity of halos.

12.7.6. 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

\[
\gamma(M) = \log_{10}(M/M_{\text{minimum}})^{-\alpha/(1+\alpha)} \quad \tag{12.70}
\]

The resulting distribution of halo masses is such that the mass of the \(i\)th halo is

\[
M_{\text{halo}, i} = \exp \left[ \ln(M_{\text{halo, min}}) + \ln \left( M_{\text{halo, max}} / M_{\text{halo, min}} \right) x_i^{1+\alpha} \right]. \tag{12.71}
\]

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.

**Halo Mass Function**

Selected with `haloMassFunctionSamplingMethod`=haloMassFunction, this method sets the sampling density function equal to the halo mass function, \(\gamma(M) = d\text{dn}(M)/d\log M\), resulting in a sample of halos representative of a volume of space.
12.8. Disk Stability/Bar Formation

Stellar Mass Function

Selected with \texttt{haloMassFunctionSamplingMethod=stellarMassFunction}, in this case the sampling density is chosen to give optimally minimal errors on the model stellar mass function (see §9.1.1 for full details). This calculation requires the observational errors on the stellar mass function to be known. A simple model of the form

\[
\sigma^2(M) = \left[ \phi_0 \left( \frac{M}{M_*} \right)^\alpha \exp \left( - \left( \frac{M}{M_*} \right)^\beta \right) \right]^2 + \sigma_0^2 \quad (12.72)
\]

is used to model the error, \( \sigma(M) \), on the observed stellar mass function as a function of stellar mass, \( M \), where \( \phi_0 = \texttt{haloMassFunctionSamplingStellarMassFunctionErrorPhi0}, M_* = \texttt{haloMassFunctionSamplingStellarMassFunctionErrorMstar}, \) and \( \sigma_0 = \texttt{haloMassFunctionSamplingStellarMassFunctionErrorConstant} \).

12.7.7. Spin Parameter Distribution

The method used to compute the distribution of dark matter halo spin parameters is selected via the \texttt{haloSpinDistributionMethod} parameter.

\textbf{Lognormal}

Selected with \texttt{haloSpinDistributionMethod=lognormal}, under this method the spin is drawn from a lognormal distribution with median \( \texttt{lognormalSpinDistributionMedian} \) and width \( \texttt{lognormalSpinDistributionSigma} \).

\textbf{Bett et al. (2007)}

Selected with \texttt{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 \( \texttt{spinDistributionBett2007Lambda0} \) and \( \texttt{spinDistributionBett2007Alpha} \) input parameters.

\textbf{Delta Function}

Selected with \texttt{haloSpinDistributionMethod=deltaFunction}, under this method the spin is drawn from a delta function distribution, \( P(\lambda) = \delta(\lambda - \lambda_0) \), where \( \lambda_0 = \texttt{deltaFunctionSpinDistributionSpin} \), i.e. a fixed value of spin equal to \( \lambda_0 \) is returned.

12.8. Disk Stability/Bar Formation

The method uses to compute the bar instability timescale for galactic disks is selected via the \texttt{barInstabilityMethod} parameter.

\textbf{12.8.1. Efstathiou, Lake & Negroponte}

Selected with \texttt{barInstabilityMethod=ELN}, this method uses the stability criterion of Efstathiou et al. [1982] to estimate when disks are unstable to bar formation:

\[
\epsilon \left( \equiv \frac{V_{\text{peak}}}{\sqrt{GM_{\text{disk}}/r_{\text{disk}}}} \right) < \epsilon_c, \quad (12.73)
\]

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

$$t_{\text{bar}} = t_{\text{disk}} \frac{\epsilon_c - \epsilon_{\text{iso}}}{\epsilon_c - \epsilon},$$

(12.74)

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.

### 12.9. Excursion Set Barrier

The functional form of the excursion set barrier is selected via the `excursionSetBarrierMethod` parameter.

#### 12.9.1. Linear Barrier

Selected with `excursionSetBarrierMethod=linear` this method assumes a barrier:

$$B(S) = B_0 + B_1 S,$$

(12.75)

where $B_0 = [\text{excursionSetBarrierConstantCoefficient}]$, and $B_1 = [\text{excursionSetBarrierLinearCoefficient}]$.

#### 12.9.2. Quadratic Barrier

Selected with `excursionSetBarrierMethod=quadratic` this method assumes a barrier:

$$B(S) = B_0 + B_1 S + B_2 S^2,$$

(12.76)

where $B_0 = [\text{excursionSetBarrierConstantCoefficient}], B_0 = [\text{excursionSetBarrierLinearCoefficient}], and B_2 = [\text{excursionSetBarrierQuadraticCoefficient}].$

#### 12.9.3. Critical Overdensity Barrier

Selected with `excursionSetBarrierMethod=criticalOverdensity` this method assumes a barrier equal to the critical linear theory overdensity for halo collapse (see §12.4.7).

### 12.10. 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.

#### 12.10.1. Linear Barrier

Selected with `excursionSetFirstCrossingMethod=linearBarrier` this method assumes the solution for a linear barrier. Specifically, the first crossing distribution is

$$f(S,t) = B(0,t) \exp(-B(S,t)^2/2S)/\sqrt{2\pi S},$$

(12.77)

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.
12.10.2. Farahi

Selected with \texttt{excursionSetFirstCrossingMethod=Faranhi} 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:

\[
1 = \int_0^S f(S')dS' + \int_{-\infty}^{B(S)} P(\delta, S)d\delta, \quad (12.78)
\]

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 \(S\):

\[
P_0(\delta, S) = \frac{1}{\sqrt{2\pi S}} \exp\left(-\frac{\delta^2}{2S}\right). \quad (12.79)
\]

Since the barrier absorbs any random walks which cross is at smaller \(S\), the actual \(P(\delta, S)\) must therefore be given by:

\[
P(\delta, S) = P_0(\delta, S) - \int_0^S f(S')P_0[\delta - B(S')], S - S']dS'. \quad (12.80)
\]

In the second term on the right hand side of eqn. (12.80) 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. (12.78):

\[
1 = \int_0^S f(S')dS' + \int_{-\infty}^{B(S)} \left[ P_0(\delta, S) - \int_0^S f(S')P_0[\delta - B(S'), S - S']dS' \right] d\delta, \quad (12.81)
\]

in general and, for the Gaussian distribution of eqn. (12.79):

\[
1 = \int_0^S f(S')dS' + \int_{-\infty}^{B(S)} \left[ \frac{1}{\sqrt{2\pi S}} \exp\left(-\frac{\delta^2}{2S}\right) - \int_0^S f(S') \frac{1}{\sqrt{2\pi (S - S')}} \exp\left(-\frac{[\delta - B(S')]^2}{2(S - S')}\right) dS' \right] d\delta. \quad (12.82)
\]

The integral over \(d\delta\) can be carried out analytically to give:

\[
1 = \int_0^S f(S')dS' + \text{erf}\left[\frac{B(S)}{\sqrt{2S}}\right] - \int_0^S f(S')\text{erf}\left[\frac{B(S) - B(S')}{\sqrt{2(S - S')}}\right] dS''. \quad (12.83)
\]

We now discretize eqn. (12.83). Specifically, we divide the \(S\) space into \(N\) intervals defined by the points:

\[
S_i = \begin{cases} 
0 & \text{if } i = 0 \\
\sum_{0}^{i-1} \Delta S_i & \text{if } i > 1.
\end{cases} \quad (12.84)
\]

Note that \(f(0) = 0\) by definition, so \(f(S_0) = 0\) always. We choose \(\Delta S_i = S_{\text{max}}/N\) (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. (12.83) gives:

\[
\int_0^S f(S')dS' = \sum_{i=0}^{i-1} \frac{f(S_i) + f(S_{i+1})}{2} \Delta S_i. \quad (12.85)
\]
and:
\[
\int_0^{S_f} f(S') \text{erf} \left[ \frac{B(S) - B(S')}{\sqrt{2}(S - S')} \right] dS' = \sum_{i=0}^{j-1} \left[ f(S_i) \text{erf} \left( \frac{B(S_j) - B(S_i)}{\sqrt{2}(S_j - S_i)} \right) + f(S_{i+1}) \text{erf} \left( \frac{B(S_j) - B(S_{i+1})}{\sqrt{2}(S_j - S_{i+1})} \right) \right] \Delta S_i.
\]

We can now rewrite eqn. (12.83) in discretized form:
\[
1 = \sum_{i=0}^{j-1} \left[ f(S_i) + f(S_{i+1}) \right] \Delta S_i + \frac{1}{2} \sum_{i=0}^{j-1} \left( f(S_i) \text{erf} \left( \frac{B(S_j) - B(S_i)}{\sqrt{2}(S_j - S_i)} \right) + f(S_{i+1}) \text{erf} \left( \frac{B(S_j) - B(S_{i+1})}{\sqrt{2}(S_j - S_{i+1})} \right) \right) \Delta S_i.
\]

Solving eqn. (12.87) for \( f(S_j) \):
\[
\left( \frac{1}{2} - \frac{1}{2} \text{erf} \left( \frac{B(S_j) - B(S_j)}{\sqrt{2}(S_j - S_j)} \right) \right) \Delta S_{j-1} f(S_j) = 1 - \sum_{i=0}^{j-2} \left( f(S_i) + f(S_{i+1}) \right) \Delta S_i - \frac{f(S_{j-1})}{2} \Delta S_{j-1} - \text{erf} \left( \frac{B(S_j)}{\sqrt{2}S_j} \right)
\]
\[+ \frac{1}{2} \sum_{i=0}^{j-2} \left( f(S_i) \text{erf} \left( \frac{B(S_j) - B(S_i)}{\sqrt{2}(S_j - S_i)} \right) + f(S_{i+1}) \text{erf} \left( \frac{B(S_j) - B(S_{i+1})}{\sqrt{2}(S_j - S_{i+1})} \right) \right) \Delta S_i + \frac{1}{2} f(S_{j-1}) \text{erf} \left( \frac{B(S_j) - B(S_{j-1})}{\sqrt{2}(S_j - S_{j-1})} \right) \Delta S_{j-1}.
\]

For all barriers that we consider:
\[
\text{erf} \left( \frac{B(S_j) - B(S_j)}{\sqrt{2}(S_j - S_j)} \right) = 0.
\]

We can then simplify eqn. (12.88):
\[
f(S_j) = \frac{2}{\Delta S_{j-1}} \left[ 1 - \sum_{i=0}^{j-2} \left( f(S_i) + f(S_{i+1}) \right) \Delta S_i - \frac{f(S_{j-1})}{2} \Delta S_{j-1} - \text{erf} \left( \frac{B(S_j)}{\sqrt{2}S_j} \right) \right]
\]
\[+ \frac{1}{2} \sum_{i=0}^{j-2} \left( f(S_i) \text{erf} \left( \frac{B(S_j) - B(S_i)}{\sqrt{2}(S_j - S_i)} \right) + f(S_{i+1}) \text{erf} \left( \frac{B(S_j) - B(S_{i+1})}{\sqrt{2}(S_j - S_{i+1})} \right) \right) \Delta S_i
\]
\[+ \frac{1}{2} f(S_{j-1}) \text{erf} \left( \frac{B(S_j) - B(S_{j-1})}{\sqrt{2}(S_j - S_{j-1})} \right) \Delta S_{j-1}.
\]

Consolidating terms in the summations:
\[
f(S_j) = \frac{2}{\Delta S_{j-1}} \left[ 1 - \text{erf} \left( \frac{B(S_j)}{\sqrt{2}S_j} \right) \right] - \sum_{i=0}^{j-1} \left( 1 - \text{erf} \left( \frac{B(S_j) - B(S_i)}{\sqrt{2}(S_j - S_i)} \right) \right) f(S_i) \Delta S_{i-1} + \Delta S_i.
\]

In the case of constant \( \Delta S_i (= \Delta S) \) this can be simplified further:
\[
f(S_j) = \frac{2}{\Delta S} \left[ 1 - \text{erf} \left( \frac{B(S_j)}{\sqrt{2}S_j} \right) \right] - \sum_{i=0}^{j-1} \left( 1 - \text{erf} \left( \frac{B(S_j) - B(S_i)}{\sqrt{2}(S_j - S_i)} \right) \right) f(S_i).
\]

In either case (i.e. eqns. 12.91 and 12.92) solution proceeds recursively: \( f(S_0) = 0 \) by definition, \( f(S_i) \) depends only on the known barrier and \( f(S_0) \), \( f(S_j) \) depends only on the known barrier and \( f(S_{j-1}) \).

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 $S$:

$$S_i = i \times \Delta S, i = 0, 1, \ldots, N, \Delta S = \frac{S}{N}. \quad (12.93)$$

Then we discretize the integral equation by Boole’s rule. The integral equation becomes a set of linear algebraic equations:

$$f(S_i) = g_1(S_i) + \frac{\Delta S}{90} \sum_{j=0}^{i-4} \left\{ 7f(S_j)g_2(S_i, S_j) + 32f(S_{j+1})g_2(S_i, S_{j+1}) + 12f(S_{j+2})g_2(S_i, S_{j+2}) + 32f(S_{j+3})g_2(S_i, S_{j+3}) + 7f(S_{j+4})g_2(S_i, S_{j+4}) \right\} \quad (12.94)$$

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:

$$f(S_i) = g_1(S_i) + \frac{4\Delta S}{90} \sum_{j=0}^{i-8} \left\{ 7f(S_j)g_2(S_i, S_j) + 32f(S_{j+1})g_2(S_i, S_{j+1}) + 12f(S_{j+2})g_2(S_i, S_{j+2}) + 32f(S_{j+3})g_2(S_i, S_{j+3}) + 7f(S_{j+4})g_2(S_i, S_{j+4}) \right\} + \frac{4\bar{g}_2(S_i)\Delta S}{90} \left( 7f(S_{i-4}) + 32f(S_{i-3}) + 12f(S_{i-2}) + 32f(S_{i-1}) + 7f(S_i) \right) \quad (12.95)$$

For $\bar{g}_2(S_i)$ we have:

$$\bar{g}_2(S_i) = \frac{1}{\delta S} \int_{S-\delta S}^{S} g_2(S, S') dS'. \quad (12.96)$$

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:

$$f(S_i) = \left( g_1(S_i) + \frac{4\Delta S}{90} \sum_{j=0}^{i-8} \left\{ 7f(S_j)g_2(S_i, S_j) + 32f(S_{j+1})g_2(S_i, S_{j+1}) + 12f(S_{j+2})g_2(S_i, S_{j+2}) + 32f(S_{j+3})g_2(S_i, S_{j+3}) + 7f(S_{j+4})g_2(S_i, S_{j+4}) \right\} + \frac{4\bar{g}_2(S_i)\Delta S}{90} \left( 7f(S_{i-4}) + 32f(S_{i-3}) + 12f(S_{i-2}) + 32f(S_{i-1}) + 7f(S_i) \right) \right) \times \left( 1 - \frac{4\bar{g}_2(S_i)\Delta S}{90} \right)^{-1}. \quad (12.97)$$
12. Physical Implementations

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:

\[
f(S_i) = \left( g_1(S_i) + \frac{\Delta S}{2} \sum_{j=0}^{m-1} \left( f(S_j)g_2(S_i, S_j) + f(S_{j+1})g_2(S_i, S_{j+1}) \right) \right) \]
\[
+ \frac{4\Delta S}{90} \sum_{i=0}^{i=8} \left\{ 7f(S_j)g_2(S_i, S_j) + 32f(S_{j+1})g_2(S_i, S_{j+1}) \right\} \]
\[
+ 12f(S_{j+2})g_2(S_i, S_{j+2}) + 32f(S_{j+3})g_2(S_i, S_{j+3}) + 7f(S_{j+4})g_2(S_i, S_{j+4}) \}
\]
\[
+ \frac{4\bar{g}_2(S_i)\Delta S}{90} \left( 7f(S_{i-4}) + 32f(S_{i-3}) + 12f(S_{i-2}) + 32f(S_{i-1}) \right) \]
\[
\times \left( 1 - \frac{4\bar{g}_2(S_i)\Delta S}{90} \right)^{-1}. \tag{12.98}
\]

Finally, we can solve the first-crossing distribution function, giving:

\[
f(S_0) = g_1(S_0) = 0 \tag{12.99}
\]
\[
f(S_1) = g_1(S_1) \left( 1 - \frac{\bar{g}_2(S_1)\Delta S}{2} \right)^{-1} \tag{12.100}
\]
\[
f(S_2) = \left( g_1(S_2) + \frac{\Delta S}{2}2\bar{g}_2(S_2)f(S_1) \right) \left( 1 - \frac{\bar{g}_2(S_2)\Delta S}{2} \right)^{-1} \tag{12.101}
\]
\[
f(S_3) = \left( g_1(S_3) + \frac{\Delta S}{2}2\bar{g}_2(S_3)(f(S_1) + f(S_2)) \right) \left( 1 - \frac{\bar{g}_2(S_3)\Delta S}{2} \right)^{-1} \tag{12.102}
\]
\[
f(S_i) = \left( g_1(S_i) + \frac{\Delta S}{2} \sum_{j=0}^{m-1} \left( f(S_j)g_2(S_i, S_j) + f(S_{j+1})g_2(S_i, S_{j+1}) \right) \right)
\]
\[
+ \frac{4\Delta S}{90} \sum_{i=0}^{i=8} \left\{ 7f(S_j)g_2(S_i, S_j) + 32f(S_{j+1})g_2(S_i, S_{j+1}) \right\} \]
\[
+ 12f(S_{j+2})g_2(S_i, S_{j+2}) + 32f(S_{j+3})g_2(S_i, S_{j+3}) + 7f(S_{j+4})g_2(S_i, S_{j+4}) \}
\]
\[
+ \frac{4\bar{g}_2(S_i)\Delta S}{90} \left( 7f(S_{i-4}) + 32f(S_{i-3}) + 12f(S_{i-2}) + 32f(S_{i-1}) \right) \]
\[
\times \left( 1 - \frac{4\bar{g}_2(S_i)\Delta S}{90} \right)^{-1}, \tag{12.103}
\]

where \( \bar{g}_2(S_1) \) is defined as:

\[
\bar{g}_2(S_1) = \int_0^{S_1} g_2(S, S')dS' \text{ if } i = 1 \tag{12.104}
\]
\[
\bar{g}_2(S_2) = \int_0^{S_2} g_2(S, S')dS' \text{ if } i = 0 \tag{12.105}
\]
\[
\bar{g}_2(S_3) = \int_0^{S_3} g_2(S, S')dS' \text{ if } i = 0 \tag{12.106}
\]
\[
\bar{g}_2(S_i) = \int_{S_{i-4}}^{S_i} g_2(S, S')dS' \text{ if } i > 3. \tag{12.107}
\]
12.11. Excursion Set Barrier Remapping

The error term for this method of discretization is:

\[ \epsilon = \frac{(\delta S)^7}{1935360} f^{(6)}(\xi), \]  

(12.108)

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:

\[ \epsilon \leq \frac{(\Delta S)^7}{118.125} f^{(6)}(\xi). \]  

(12.109)

Since there are \( N/4 \) intervals, the maximum deviation from the real value of the function is:

\[ \epsilon \leq \sum_{i=1}^{N/4} \frac{N}{4} \frac{(\Delta S)^7}{118.125} f^{(6)}(\xi_i) \leq N \frac{(\Delta S)^7}{472.5} f^{(6)}(\xi), \]  

where \( f^{(6)}(\xi) \) is the absolute maximum of the sixth derivative in the domain, \([0, S]\).

12.11. 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.

12.11.1. Null Remapping

Selected with `excursionSetBarrier(Rates)RemapMethods=null` this method leaves the barrier unmodified.

12.11.2. Scale Remapping

Selected with `excursionSetBarrier(Rates)RemapMethods=scale` this method multiplies the barrier by \([\text{excursionSetBarrierRemapScalingFactor}]\).

12.11.3. 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]:

\[ B(S) \rightarrow \sqrt{A}B(S) \left( 1 + b \left[ \frac{S}{AB^2(S)} \right]^c \right), \]  

(12.111)

where \( A = 0.707 \), \( b = 0.5 \), and \( c = 0.6 \).

12.12. Galactic Structure

The algorithm to be used when solving for galactic structure (specifically, finding radii of galactic components) is selected via the `galacticStructureRadiusSolverMethod` parameter.
12. Physical Implementations

12.12.1. Fixed

Selected with `galacticStructureRadiusSolverMethod=fixed` this method determines the sizes of galactic components by assuming that radius equals

\[ r = f_r \lambda r_{\text{vir}} \quad (12.112) \]

where \( r_{\text{vir}} \) is the virial radius of the node, \( \lambda \) is its spin parameter and \( f_r = \text{[galacticStructureRadiiFixedFactor]} \) is a parameter.

12.12.2. 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

\[ r = r_{\text{vir}} j / j_{\text{vir}} \quad (12.113) \]

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.

12.12.3. 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

\[ j = \sqrt{G M(r)} r, \quad (12.114) \]

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).

12.12.4. 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 §12.13). 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).

12.13. Galactic Structure Initial Radii

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.
12.13.1. Static

Selected with `galacticStructureRadiusSolverInitialRadiusMethod=static` this method assumes that the dark matter halo is static and so sets the initial radius equal to the final radius.

12.13.2. Adiabatic

Selected with `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 `adiabaticContractionGnedinA` and `adiabaticContractionGnedinOmega` respectively.

Given the final radius, $r_f$, the corresponding initial radius, $r_i$, is found by solving:

$$f_i M_{\text{total},0}(\bar{r}_i) r_i = f_i M_{\text{total},0}(\bar{r}_i) r_i + V_b^2(\bar{r}_i) \bar{r}_i r_i / G,$$

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\(^1\), $f_i = (\Omega_M - \Omega_b)/\Omega_M + M_{\text{satellite,baryonic}}/M_{\text{total}}$, $M_{\text{satellite,baryonic}}$ is the baryonic mass in any satellite halos, $M_{\text{total}}$ is the node mass, and

$$\frac{\bar{r}}{r_{\text{vir}}} = A \left( \frac{r}{r_{\text{vir}}} \right)^{\omega},$$

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, $dr_f/dd_i \equiv r'_i$, is found by taking the derivative of eqn. (12.115) to give:

$$f_i M_{\text{total},0}(\bar{r}_i) r'_i + f_i 4\pi \bar{r}_i^2 \rho_{\text{total},0}(\bar{r}_i) \frac{d\bar{r}_i}{dr_i} r'_i$$

$$= f_i M_{\text{total},0}(\bar{r}_i) + f_i 4\pi \bar{r}_i^2 \rho_{\text{total},0}(\bar{r}_i) \frac{d\bar{r}_i}{dr_i} r'_i$$

$$+ V_b^2(\bar{r}_i) \bar{r}_i / G + V_b^2(\bar{r}_i) \frac{d\bar{r}_i}{dr_i} r_i / G + \frac{dV_b^2}{dr_i}(\bar{r}_i) \frac{d\bar{r}_i}{dr_i} r_i / G,$$

where

$$\frac{d\bar{r}}{dr} = A \left( \frac{r}{r_{\text{vir}}} \right)^{\omega-1},$$

and which can then be solved numerically for $r'_i$.


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.


The movement of mass elements in the merging galaxies is determined by the `satelliteMergingMassMovementsMethod` parameter.

\(^1\)In 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 $f_i < 1$.\[205\]
12. 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 `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 `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 \( f_{\text{gas,crit}} = [\text{burstCriticalGasFraction}] \).

12.14.2. 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).

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

\[
\frac{(M_1 + M_2)^2}{r_{\text{new}}} = \frac{M_1^2}{r_1} + \frac{M_2^2}{r_2} + \frac{\text{f}_{\text{orbit}} M_1 M_2}{c \cdot r_1 + r_2},
\]

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, \(r_1\) 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 \(\text{f}_{\text{orbit}}\) depends on the orbital parameters of the galaxy pair. For example, a value of \(\text{f}_{\text{orbit}} = 1\) 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, \(\text{f}_{\text{boost}}\), to the specific angular momentum of each component of each merging galaxy is computed:

\[
\text{f}_{\text{boost}} = \frac{j}{\sqrt{GM r_{1/2}}},
\]

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:

\[
\text{j}_{\text{new}} = \langle \text{f}_{\text{boost}} \rangle \text{r}_{\text{new}} \text{V}_{\text{new}},
\]

where

\[
\text{V}_{\text{new}}^2 = \frac{G(M_1 + M_2)}{r_{\text{new}}}.
\]

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

\[
\frac{(M_1 + M_2)^2}{r_{\text{new}}} = \left[ \frac{M_1^2}{r_1} + \frac{M_2^2}{r_2} + \frac{\text{f}_{\text{orbit}} M_1 M_2}{c \cdot r_1 + r_2} \right] (1 + \text{f}_{\text{gas}} C_{\text{rad}}),
\]

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 \(\text{f}_{\text{orbit}}\) depends on the orbital parameters (see §12.14.1) not all mass may be placed into the spheroid component of the remnant.

\(2\)Depending on the merging rules (see §12.14.1) not all mass may be placed into the spheroid component of the remnant.

\(3\)In practice, \textsc{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.
12. Physical Implementations

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. (12.123) 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 §12.14.2).

12.14.3. Progenitor Properties

The method used to calculate the properties of merger progenitor galaxies is selected by the satelliteMergingRemnantProgenitorPropertiesMethod parameter.

Cole et al. (2000)

Selected using satelliteMergingRemnantProgenitorPropertiesMethod=Cole2000, this method uses the algorithms of Cole et al. [2000] to compute progenitor properties. Masses of progenitors are set to

$$M_{\text{host|satellite}} = \sum_{i=\text{disk/spheroid}} \sum_{j=\text{stars/gas}} M_{i,j},$$

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

$$M_{\text{spheroid host|satellite}} = \sum_{i=\text{disk/spheroid}} \sum_{j=\text{stars/gas}} M_{i,j} \delta_{i,j},$$

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:

$$\sum_{i=\text{disk/spheroid}} \sum_{j=\text{stars/gas}} M_{i,j}(r) \delta_{i,j} = \frac{1}{2} \sum_{i=\text{disk/spheroid}} \sum_{j=\text{stars/gas}} M_{i,j} \delta_{i,j},$$

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

$$f_{\text{AM host|satellite}} = \frac{1}{M_{\text{spheroid host|satellite}}} \sum_{i=\text{disk/spheroid}} \sum_{j=\text{stars/gas}} M_{i,j} \frac{J_{i,j}}{G M_{\text{remnant}}^{3/2} r_{1/2}^{1/2} \delta_{i,j}},$$

where $J_{i,j}$ is the angular momentum or pseudo-angular momentum of mass type $j$ in component $i^4$.  

---

*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 $j = \sqrt{GM_{\text{remnant}}^{3/2} r_{1/2}^{1/2}}$ (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 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 GalactICUS.

208
Standard

Selected using \texttt{satelliteMergingRemnantProgenitorPropertiesMethod=standard}, this is the standard method to compute progenitor properties. Masses of progenitors are set to

\begin{equation}
M_{\text{host|satellite}} = \sum_{i=\text{disk|spheroid}} \sum_{j=\text{stars|gas}} M_{i,j},
\end{equation}

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

\begin{equation}
M_{\text{spheroid host|satellite}} = \sum_{i=\text{disk|spheroid}} \sum_{j=\text{stars|gas}} M_{i,j}\delta_{i,j},
\end{equation}

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

\begin{equation}
r_{\text{host|satellite}} = \frac{1}{M_{\text{spheroid host|satellite}}} \sum_{i=\text{disk|spheroid}} \sum_{j=\text{stars|gas}} M_{i,j}r_{1/2}^i j \delta_{i,j}.
\end{equation}

Finally, the angular momentum factor is set to

\begin{equation}
J_{\text{AM host|satellite}} = \frac{1}{M_{\text{spheroid host|satellite}}} \sum_{i=\text{disk|spheroid}} \sum_{j=\text{stars|gas}} M_{i,j} \frac{J_{i,j}}{GM_{i,j}^2 r_{1/2}^i j} \delta_{i,j},
\end{equation}

where $J_{i,j}$ is the angular momentum or pseudo-angular momentum of mass type $j$ in component $i$.

12.15. Hot Halo Density Profile

The algorithm to be used when determining the hot halo density profile is selected via the \texttt{hotHaloDensityMethod} parameter.

12.15.1. Cored Isothermal

Selected with \texttt{hotHaloDensityMethod=coredIsothermal} this method adopts a spherically symmetric cored-isothermal density profile for the hot halo. Specifically,

\begin{equation}
\rho_{\text{hothalo}}(r) \propto \left[r^2 + r_{\text{core}}^2\right]^{-2},
\end{equation}

where the core radius, $r_{\text{core}}$, is set using the selected cored isothermal core radius method (see §12.16). The profile is normalized such that the current mass in the hot gas profile is contained within the outer radius of the hot halo, $r_{\text{hot,outer}}$. The rotation normalization, i.e. the constant $A$ in the relation $A = V_{\text{rotation}}/\langle j \rangle$ where $V_{\text{rotation}}$ is an assumed constant with radius rotation speed and $\langle j \rangle$ is the mean specific angular momentum of the gas, is given by

\begin{equation}
A = r_{\text{hot,outer}}^{-1} \frac{1 - x_{\text{core}} \arctan(x_{\text{core}}^{-1})}{1/2 + x_{\text{core}}^2 \ln(x_{\text{core}}/[1 + x_{\text{core}}^2])^{1/2}},
\end{equation}

where $x_{\text{core}} = r_{\text{core}}/r_{\text{hot,outer}}$.

Finally, only the second and third radial moments, $I_2$ and $I_3$ respectively, are supported, for which

\begin{equation}
I_2 = x_{\text{core}} - \tan^{-1}(x_{\text{core}}),
\end{equation}

and

\begin{equation}
I_3 = \frac{1}{2} \left[x_{\text{core}}^2 - \log(1 + x_{\text{core}}^2)\right].
\end{equation}
12. Physical Implementations

12.15.2. Null

Selected with `hotHaloDensityMethod=null` this method assumes no hot halo density profile. It is useful, for example, when performing dark matter-only calculations.

12.16. Hot Halo Density Profile: Cored Isothermal Core Radius

The algorithm to be used when determining the core radius of cored isothermal hot halo density profiles is selected via the `hotHaloCoredIsothermalCoreRadiiMethod` parameter.

12.16.1. Growing Core

Selected with `hotHaloCoredIsothermalCoreRadiiMethod=growingCore`, this method implements a core radius equal to a fraction \( \frac{\text{isothermalCoreRadius}}{\text{ScaleRadius}} \) 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 \( \frac{\text{isothermalCoreRadius}}{\text{VirialRadiusMaximum}} \) times the node’s virial radius.

12.16.2. Virial Radius Fraction

Selected with `hotHaloCoredIsothermalCoreRadiiMethod=virialRadiusFraction`, this method implements a core radius equal to a fraction \( \frac{\text{isothermalCoreRadius}}{\text{VirialRadius}} \) of the node’s virial radius.

12.17. 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 `hotHaloRamPressureForceMethod` parameter.

12.17.1. Null

Selected with `hotHaloRamPressureForceMethod=null` this method assumes a zero ram pressure force due to the hot halo.

12.17.2. Font et al. (2008)

Selected with `hotHaloRamPressureForceMethod=Font2008` this method computes the ram pressure force using the algorithm of Font et al. [2008]. Specifically, the ram pressure force is

\[
\mathcal{F}_{\text{ram, hot, host}} = \rho_{\text{hot, host}}(r_{\text{peri}}) v^2(r_{\text{peri}}),
\]

(12.137)

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.

12.18. 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.
12.18.1. 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.

12.18.2. Font et al. (2008)

Selected with `hotHaloRamPressureStrippingMethod=Font2008` this method computes the ram pressure stripping radius using the algorithm of Font et al. [2008]. Specifically, the radius, \( r_{\text{rp}} \), is computed as the solution of

\[
\alpha_{\text{rp}} \frac{G M_{\text{satellite}}(r_{\text{rp}}) \rho_{\text{hot,satellite}}(r_{\text{rp}})}{r_{\text{rp}}} = F_{\text{ram,hot,host}},
\]

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 §12.17). The parameter \( \alpha_{\text{rp}} = [\text{ramPressureStrippingFormFactor}] \) is a geometric factor of order unity.

12.19. Hot Halo Temperature Profile

The algorithm to be used when determining the hot halo temperature profile is selected via the `hotHaloTemperatureMethod` parameter.

12.19.1. Virial Temperature

Selected with `hotHaloTemperatureMethod=virial` this method assumes an isothermal halo with a temperature equal to the virial temperature of the halo.

12.20. 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.

12.20.1. 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).
12. Physical Implementations

12.20.2. Initial Mass Functions

A variety of different IMF s 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 $\text{imf[imfName]RecycledInstantaneous}$ and $\text{imf[imfName]YieldInstantaneous}$ 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]:

$$\phi(M) \propto M^{-1} \text{ for } 0.15M_\odot < M < 125M_\odot$$ (12.139)

**Chabrier**

The Chabrier IMF is defined by [Chabrier, 2001]:

$$\phi(M) \propto \begin{cases} M^{-1} \exp(-[\log_{10}(M/M_c)/\sigma_c]^2/2) & \text{ for } 0.1M_\odot < M < 1M_\odot \\ M^{-2.3} & \text{ for } 1M_\odot < M < 125M_\odot \\ 0 & \text{ otherwise} \end{cases}$$ (12.140)

where $\sigma_c = 0.69$ and $M_c = 0.08M_\odot$.

**Kennicutt**

The Kennicutt IMF is defined by [Kennicutt, 1983]:

$$\phi(M) \propto \begin{cases} M^{-1.25} & \text{ for } 0.10M_\odot < M < 1.00M_\odot \\ M^{-2.00} & \text{ for } 1.00M_\odot < M < 2.00M_\odot \\ M^{-2.30} & \text{ for } 2.00M_\odot < M < 125M_\odot \\ 0 & \text{ otherwise} \end{cases}$$ (12.141)

**Kroupa**

The Kroupa IMF is defined by [Kroupa, 2001]:

$$\phi(M) \propto \begin{cases} M^{-0.3} & \text{ for } 0.01M_\odot < M < 0.08M_\odot \\ M^{-1.8} & \text{ for } 0.08M_\odot < M < 0.5M_\odot \\ M^{-2.7} & \text{ for } 0.5M_\odot < M < 1M_\odot \\ M^{-2.3} & \text{ for } 1M_\odot < M < 125M_\odot \\ 0 & \text{ otherwise} \end{cases}$$ (12.142)

**Miller-Scalo**

The Miller-Scalo IMF is defined by [Miller and Scalo, 1979]:

$$\phi(M) \propto \begin{cases} M^{-1.25} & \text{ for } 0.10M_\odot < M < 1.00M_\odot \\ M^{-2.00} & \text{ for } 1.00M_\odot < M < 2.00M_\odot \\ M^{-2.30} & \text{ for } 2.00M_\odot < M < 10.0M_\odot \\ M^{-3.30} & \text{ for } 10.0M_\odot < M < 125M_\odot \\ 0 & \text{ otherwise} \end{cases}$$ (12.143)
12.21. Intergalactic Medium State

**Piecewise Power-law**

Arbitrary piecewise power-law IMFs can be defined using the `PiecewisePowerLaw` method. The IMF will be constructed such that:

\[
\phi(M) \propto M^{\alpha_i} \text{ if } M_i \leq M < M_{i+1},
\]

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]:

\[
\phi(M) \propto \begin{cases} 
M^{-2.35} & \text{for } 0.1M_\odot < M < 125M_\odot \\
0 & \text{otherwise.}
\end{cases}
\]

(12.145)

**Scalo**

The Scalo IMF is defined by [Scalo, 1986]:

\[
\phi(M) \propto \begin{cases} 
M^{+1.60} & \text{for } 0.10M_\odot < M < 0.18M_\odot \\
M^{-1.01} & \text{for } 0.18M_\odot < M < 0.42M_\odot \\
M^{-2.75} & \text{for } 0.42M_\odot < M < 0.62M_\odot \\
M^{-2.08} & \text{for } 0.62M_\odot < M < 1.18M_\odot \\
M^{-3.50} & \text{for } 1.18M_\odot < M < 3.50M_\odot \\
M^{-2.63} & \text{for } 3.50M_\odot < M < 125M_\odot \\
0 & \text{otherwise.}
\end{cases}
\]

(12.146)

12.21. Intergalactic Medium State

The thermal and ionization state of the intergalactic medium is implemented by the algorithm selected using the `intergalaticMediumStateMethod` parameter.

12.21.1. **RecFast**

Selected using `intergalaticMediumStateMethod=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.

12.21.2. **File**

Selected using `intergalaticMediumStateMethod=file`, this method reads the state of the intergalactic medium from a file and interpolates in the tabulated results. The format of the file is specified in §16.4.1.

12.22. Chemical State

The chemical state of gas is implemented by the algorithm selected using the `ionizationStateMethod` parameter.
12. Physical Implementations

12.22.1. Atomic Collisional Ionization Equilibrium Using CLOUDY

Selected using ionizationStateMethod=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.

12.22.2. Collisional Ionization Equilibrium From File

Selected using ionizationStateMethod=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 §16.4.1. The chemical state is assumed to be computed under conditions of collisional ionization equilibrium and therefore densities scale as \( \rho \). Optional \( \text{H}^0 \) and \( \text{H}^+ \) densities, if present in the file, will be read and used when returning the densities of “chemical” species.

12.23. Merger Tree Construction

Merger trees are “constructed\(^5\)” by the method specified by the mergerTreeConstructMethod parameter.

12.23.1. Read From File

Selected with mergerTreeConstructMethod= read, 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.

12.23.2. Build

Selected with mergerTreeConstructMethod= build, this method first creates a distribution of tree root halo masses and then builds a merger tree using the algorithm specified by the mergerTreeBuildMethod parameter.

If [mergerTreeBuildTreesHaloMassDistribution]=read, then these masses will be read from a file specified by [mergerTreeBuildTreeMassesFile]. Otherwise, the root halo masses are selected to range between mergerTreeBuildHaloMassMinimum and mergerTreeBuildHaloMassMaximum with mergerTreeBuildTreesPerDecade trees per decade of root halo mass on average. Trees are rooted at mergerTreeBuildTreesBaseRedshift and tree building will begin with the mergerTreeBuildTreesBeginAtTree\(^6\)th tree. The root halo masses are drawn from the density function specified by the halo mass function sampling density method (see §12.7.6). The distribution of \( x \) is determined by the input parameter mergerTreeBuildTreesHaloMassDistribution with options:

- uniform \( x \) is distributed uniformly between 0 and 1;
- quasi \( x \) is distributed using a quasi-random sequence.

\(^5\)By “construct” we mean any process of creating a representation of a merger tree within 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.
In the case of reading root halo masses from a file, the file should be an XML file with the following form:

```xml
<mergerTrees>
  <treeRootMass>13522377303.5998</treeRootMass>
  <treeRootMass>19579530191.8709</treeRootMass>
  <treeRootMass>21061025282.9613</treeRootMass>
  ...
</mergerTrees>
```

where each `treeRootMass` element gives the mass (in Solar masses) of the root halo of a tree to generate.

### 12.23.3. 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>
      <abundancesGas>
        <metals>1.0e9</metals>
        <Fe>1.0e9</Fe>
      </abundancesGas>
    </disk>
  </node>
  <node>
    <index>1</index>
    <parent>-1</parent>
</initialConditions>
```
12. 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).

12.23.4. 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 §16.4.1). 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 mergerTreeHaloMassDeclineFactor at each new node until a specified mergerTreeHaloMassResolution (in units of $M_\odot$) is reached.

12.23.5. State Restore

Selected with 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 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_Tree_State_Store(thisTree,'storedTree.dat')
       stop 'tree internal state was stored'
   end if
   ```

3. Run the model ensuring that [stateFileRoot] is set to a suitable file root name to allow the internal state of GALACTICUS to be stored;

4. Remove the code inserted above and recompile;

5. Run GALACTICUS with an input parameter file identical to the one used previously except with [mergerTreeConstructMethod]=stateRestore, [stateFileRoot] removed, [stateRetrieveFileRoot] set to the value previously used for [stateFileRoot] and [mergerTreeStateStoreFile]=storedTree.dat.

This should restore the tree and the internal state of 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.

### 12.24. Merger Tree Branching

The method to be used for computing branching probabilities in merger trees is specified by the `treeBranchingMethod` parameter.

#### 12.24.1. 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 the first order expansion of the merging rate that is assumed is accurate.

#### 12.24.2. 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 §16.4.1). Specifically, the branching probability per unit time is:

\[
\frac{df}{dt} = \frac{d}{d\omega} \int_{M_{\text{min}}}^{M_2^2} M \left( \frac{df}{dt} dS \right) \left| \frac{d}{d\omega} G(\omega, \sigma(M), \sigma(M')) \right| dM',
\]

(12.147)
12. Physical Implementations

where $\omega = \delta_c / D(t)$. The rate of accretion of mass in halos below the resolution limit of the merger tree is

$$\frac{dR}{dt} = \frac{dt}{d\omega} \int_{M_{\text{min}}}^{M} \frac{df}{dS} \frac{dM'}{d\omega} G[\omega, \sigma(M), \sigma(M')] dM'. \quad (12.148)$$

In the above, $G[\omega, \sigma(M), \sigma(M')]$ is a modification to the merger rate as computed by the selected treeBranchingModifierMethod (see §16.4.1). If [generalizedPressSchechterSmoothAccretion]=true then smooth accretion (i.e. accretion of matter not in dark matter halos) is accounted for at the rate:

$$\frac{dR_s}{dt} = \frac{dt}{d\omega} G[\omega, \sigma_{\text{max}}, \sigma(M')] \frac{d\tilde{f}}{dt}, \quad (12.149)$$

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.

12.25. Merger Tree Branching Modifier

The method to be used for modifying branching probabilities in merger trees is specified by the treeBranchingModifierMethod parameter.

12.25.1. 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:

$$G(\delta_p, \sigma_c, \sigma_p) = G_0 \left( \frac{\sigma_p}{\sigma_c} \right)^{\gamma_1} \left( \frac{\delta_p}{\sigma_p} \right)^{\gamma_2} \quad (12.150)$$

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: $G_0$ = [modifiedPressSchechterG0], $\gamma_1$ = [modifiedPressSchechterGamma1], and $\gamma_2$ = [modifiedPressSchechterGamma2].

12.26. 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.

mergerTreeBuildCole2000MassResolution The minimum halo mass (in $M_\odot$) that the algorithm will follow. Mass accretion below this scale is treated as smooth accretion and branches are truncated once they fall below this mass.

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.

12.27. Merger Tree Pre-evolution Processing

Arbitrary processing of merger trees prior to their evolution can be carried out using the `mergerTreePreEvolveTask` directive (see §16.4.3). Currently defined tasks are defined below.

12.27.1. 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`.

12.27.2. 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:

[mergerTreeRegridStartExpansionFactor] The smallest expansion factor in the array;

[mergerTreeRegridEndExpansionFactor] The largest expansion factor in the array;

[mergerTreeRegridCount] The number of timesteps in the array;

[mergerTreeRegridSpacing] The spacing of the timesteps. Two options are available: `linear` will space timesteps uniformly in expansion factor, while `logarithmic` will space timesteps uniformly in the logarithm of expansion factor.

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.

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.
12. Physical Implementations

12.27.3. 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, \text{massAccretionHistories}, is made in the \text{Galacticus} output file. This group will contain a subgroup for each merger tree (\text{mergerTreeN} where \( N \) is the merger tree index) within which three datasets, \text{nodeIndex}, \text{nodeTime} and \text{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.

12.27.4. 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.

12.27.5. 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.)

12.28. 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.

12.28.1. Null

Selected with \[\text{chemicalReactionRatesMethods} = \text{null} \] this is a null method which does not set any rates.

12.28.2. Hydrogen Network

Selected with \[\text{chemicalReactionRatesMethods} = \text{hydrogenNetwork} \] this method computes rates using the network of reactions and fitting functions from Abel et al. [1997] and Tegmark et al. [1997]. The parameter \[\text{hydrogenNetworkFast} \] controls the approximations made. If set \text{true} then \( \text{H}^- \) is assumed to be at equilibrium abundance, \( \text{H}_2^+ \) reactions are ignored and other slow reactions are ignored (see Abel et al. 1997).

12.29. Ram Pressure Induced Mass Loss Rates in Disks

The methods for computing ram pressure induced rates of mass loss in disks are selected via the \[\text{ramPressureStrippingMassLossRateDisksMethod} \] parameter.

---

8The 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.
12.29.1. Simple

Selected with `ramPressureStrippingMassLossRateDisksMethod=simple` this method computes the mass loss rate to be:

\[
\dot{M}_{\text{gas, disk}} = \min \left( \frac{F_{\text{hot, host}}}{2\pi G \Sigma_{\text{gas}}(r_{\text{half}}) \Sigma_{\text{total}}(r_{\text{half}})}, R_{\text{maximum}} \right) \frac{M_{\text{gas, disk}}}{\tau_{\text{dyn, disk}}}.
\]  

(12.151)

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 §12.17), \(\Sigma_{\text{gas}}(r)\) 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}} = [\text{ramPressureStrippingMassLossRateDiskSimpleFractionalRateMaximum}\] controls the maximum allowed rate of mass loss.

12.29.2. Null

Selected with `ramPressureStrippingMassLossRateDisksMethod=null` this method assumes zero mass loss rate always.

12.30. Star Formation Rate Surface Densities

The method for computing surface densities of star formation rate in disks is selected via `starFormationRateSurfaceDensityDisksMethod`.

12.30.1. Kennicutt-Schmidt

Selected with `starFormationRateSurfaceDensityDisksMethod=Kennicutt-Schmidt` this method assumes that the Kennicutt-Schmidt law holds [Schmidt, 1959, Kennicutt, 1998]:

\[
\dot{\Sigma}_* = A \left( \frac{\Sigma_H}{M_\odot \text{pc}^{-2}} \right)^N,
\]  

(12.152)

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

\[
\dot{\Sigma}_* = \begin{cases} 
A \left( \frac{\Sigma_H}{M_\odot \text{pc}^{-2}} \right)^N & \text{if } \Sigma_{\text{gas, disk}} > \Sigma_{\text{crit}} \\
A \left( \frac{\Sigma_H}{M_\odot \text{pc}^{-2}} \right)^N \left( \frac{\Sigma_{\text{gas, disk}}}{\Sigma_{\text{crit}}} \right)\alpha & \text{otherwise.}
\end{cases}
\]  

(12.153)

Here, \(\alpha = [\text{starFormationKennicuttSchmidtExponentTruncated}]\) and \(\Sigma_{\text{crit}}\) is a critical surface density for star formation which we specify as

\[
\Sigma_{\text{crit}} = \frac{q_{\text{crit}} \kappa \sigma_{\text{gas}}}{\pi G},
\]  

(12.154)

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 \([\text{velocityDispersionDiskGas}]\) and that the disk has a flat rotation curve such that \(\kappa = \sqrt{2V/R}\).
12. Physical Implementations

12.30.2. Extended Schmidt

Selected with \texttt{starFormationRateSurfaceDensityDisksMethod=extendedSchmidt} this method assumes that the extended Schmidt law holds [Shi et al., 2011]:

\[ \dot{\Sigma}_\star = A \left( x_H \frac{\Sigma_{\text{gas}}}{M_\odot \text{pc}^{-2}} \right)^{N_1} \left( \frac{\Sigma_\star}{M_\odot \text{pc}^{-2}} \right)^{N_2} \]  \hspace{1cm} (12.155)

where \( A = \text{[starFormationExtendedSchmidtNormalization]} \), \( N_1 = \text{[starFormationExtendedSchmidtGasExponent]} \) and \( N_2 = \text{[starFormationExtendedSchmidtStarExponent]} \) are parameters.

12.30.3. Blitz-Rosolowsky

Selected with \texttt{starFormationRateSurfaceDensityDisksMethod=Blitz-Rosolowsky2006} this method assumes that the star formation rate is given by [Blitz and Rosolowsky, 2006]:

\[ \dot{\Sigma}_\star(R) = \nu_{\text{SF}}(R) \Sigma_{\text{H}_2,\text{disk}}(R), \]  \hspace{1cm} (12.156)

where \( \nu_{\text{SF}} \) is a frequency given by

\[ \nu_{\text{SF}}(R) = \nu_{\text{SF,0}} \left[ 1 + \left( \frac{\Sigma_{\text{HI}}}{\Sigma_0} \right)^q \right], \]  \hspace{1cm} (12.157)

where \( q = \text{[surfaceDensityExponentBlitzRosolowsky]} \) and \( \Sigma_0 = \text{[surfaceDensityCriticalBlitzRosolowsky]} \) are parameters and the surface density of molecular gas \( \Sigma_{\text{H}_2} = (P_{\text{ext}}/P_0)^\alpha \Sigma_{\text{HI}}, \) 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

\[ P_{\text{ext}} \approx \frac{\pi}{2} G \Sigma_{\text{gas}} \left[ \Sigma_{\text{gas}} + \left( \frac{\sigma_{\text{gas}}}{\sigma_\star} \right) \Sigma_\star \right] \]  \hspace{1cm} (12.158)

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

\[ \sigma_\star = \sqrt{\pi G \Sigma_{\text{gas}}} \]  \hspace{1cm} (12.159)

where we assume \( h_\star / R_{\text{disk}} = \text{[heightToRadialScaleDiskBlitzRosolowsky]} \).

12.30.4. Krumholz-McKee-Tumlinson

Selected with \texttt{starFormationRateSurfaceDensityDisksMethod=KMT09} this method assumes that the star formation rate is given by [Krumholz et al., 2009]:

\[ \dot{\Sigma}_\star(R) = \nu_{\text{SF}} f_{\text{H}_2}(R) \Sigma_{\text{HI, disk}}(R) \left\{ \frac{(\Sigma_{\text{HI}}/\Sigma_0)^{-1/3}}{(\Sigma_{\text{HI}}/\Sigma_0)^{1/3}}, \begin{array}{ll} & \text{if } \Sigma_{\text{HI}}/\Sigma_0 \leq 1 \\ \Sigma_{\text{HI}}/\Sigma_0 \geq 1 \\ & \end{array} \right. \]  \hspace{1cm} (12.160)

where \( \nu_{\text{SF}} = \text{[starFormationFrequencyKMT09]} \) is a frequency and \( \Sigma_0 = 85 M_\odot \text{pc}^{-2} \). The molecular fraction is given by

\[ f_{\text{H}_2} = 1 - \left( 1 + \left[ \frac{3 \delta}{4(1 + \delta)} \right]^{-5} \right)^{-1/5} \]  \hspace{1cm} (12.161)

where

\[ \delta = 0.0712 \left[ 0.1 \lambda^{-1} + 0.675 \right]^{-2.8} \]  \hspace{1cm} (12.162)
and
\[ s = \frac{\ln(1 + 0.6\chi + 0.01\chi^2)}{0.04\Sigma_{\text{comp},0}Z'}, \quad (12.163) \]
with
\[ \chi = 0.77 \left[ 1 + 3.1Z'^{0.365} \right], \quad (12.164) \]
and \( \Sigma_{\text{comp},0} = c\Sigma_{\text{HI}}/M_\odot \text{pc}^{-2} \) where \( c = \text{[molecularComplexClumpingFactorKMT09]} \) is a density enhancement factor relating the surface density of molecular complexes to the gas density on larger scales.
Alternatively, if \( \text{[molecularFractionFastKMT09]} \) is set to true, the molecular fraction will be computed using the faster (but less accurate at low molecular fraction) formula
\[ f_{H_2} = 1 - \frac{3s/4}{(1 + s/4)}. \quad (12.165) \]

12.31. Star Formation Timescales

The methods for computing star formation timescales in disks and spheroids are selected via the \text{starFormationTimescaleDisksMethod} and \text{starFormationTimescaleSpheroidsMethod} respectively.

12.31.1. Fixed

Selected with \text{starFormationTimescaleDisksMethod=fixed} this method assumes a fixed timescale for star formation \( \text{[starFormationTimescaleDisksFixedTimescale]} \) (in Gyr).

12.31.2. Halo Scaling

Selected with \text{starFormationTimescaleDisksMethod=haloScaling} this method assumes a timescale for star formation of
\[ \tau_* = \tau_{*,0} \left( \frac{V_{\text{vir}}}{200 \text{km/s}} \right)^{\alpha_*} (1 + z)^{\beta_*}, \quad (12.166) \]
where \( \tau_{*,0} = \text{[starFormationTimescaleDisksHaloScalingTimescale]} \), \( \alpha_* = \text{[starFormationTimescaleDisksHaloScalingVirialVelocityExponent]} \) and \( \beta_* = \text{[starFormationTimescaleDisksHaloScalingRedshiftExponent]} \).

12.31.3. Dynamical Time

Selected with \text{starFormationTimescale[Disks|Spheroids]Method=dynamicalTime} this method computes the star formation timescale to be:
\[ \tau_* = \epsilon_*^{-1} \tau_{\text{dynamical}} \left( \frac{V}{200 \text{km/s}} \right)^{\alpha_*}, \quad (12.167) \]
where \( \epsilon_* = \text{[starFormation[Disks|Spheroids]Efficiency]} \) 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 \text{starFormation[Disks|Spheroids]MinimumTimescale} (in Gyr).
12. Physical Implementations

12.31.4. Integrated Surface Density

Selected with starFormationTimescaleDisksMethod=integratedSurfaceDensity this method computes the star formation timescale to be:

$$
\tau_* = \frac{M_{\text{cold}}}{\int_0^\infty 2\pi r \dot{\Sigma}_* (r) \, dr},
$$

(12.168)

where $\dot{\Sigma}_*(r)$ is the surface density of star formation rate (see §12.30).

12.31.5. Baugh et al. (2005)

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:

$$
\tau_* = \tau_0 (V_{\text{disk}}/200 \text{km/s})^\alpha a^\beta
$$

(12.169)

where $\tau_0 = [\text{starFormationDiskTimescale}]$, $\alpha = [\text{starFormationDiskVelocityExponent}]$ and $\beta = [\text{starFormationExpansionExponent}]$.

12.32. 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.

12.32.1. 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 $\dot{M}_* = (1 - R)\phi$, 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

$$
\dot{M}_{\text{fuel}, Z} = -(1 - R) Z_{\text{fuel}} \phi + p \phi.
$$

(12.170)

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.

12.32.2. 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:

$$
\dot{M}_* = \phi - \int_0^t \phi(t') \dot{R}(t-t'; Z_{\text{fuel}}(t')) \, dt',
$$

(12.171)

$$
\dot{M}_{\text{fuel}} = -\phi + \int_0^t \phi(t') \dot{R}(t-t'; Z_{\text{fuel}}(t')) \, dt',
$$

(12.172)
\[ \dot{M}_{*,Z} = Z_{\text{fuel}} \dot{\phi} - \int_0^t \dot{\phi}(t') Z_{\text{fuel}}(t'') \dot{R}(t - t''; Z_{\text{fuel}}(t'')) \, dt', \tag{12.173} \]
\[ \dot{M}_{\text{fuel},Z} = -Z_{\text{fuel}} \dot{\phi} + \int_0^t \dot{\phi}(t') \left( Z_{\text{fuel}}(t'') \dot{R}(t - t''; Z_{\text{fuel}}(t'')) + \dot{p}(t - t''; Z_{\text{fuel}}(t'')) \right) \, dt', \tag{12.174} \]
\[ \tag{12.175} \]

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.

### 12.33. 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 §16.4.3).

#### 12.33.2. File

Selected with `stellarPopulationSpectraMethod=file` this method reads stellar population spectra from an HDF5 file, with format described in §16.4.1.

### 12.34. 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]`.

#### 12.34.1. Meiksin (2006) IGM Attenuation

Selected with `stellarPopulationSpectraPostprocess<setName>Method=Meiksin2006` this method postprocesses spectra through absorption by the IGM using the results of Meiksin [2006].

#### 12.34.2. 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].
12. Physical Implementations

12.34.3. Lyman-continuum Suppression

Selected with `stellarPopulationSpectraPostprocess<setName>Method=lymanContinuumSuppression` this method suppresses all emission in the Lyman continuum.

12.34.4. Recent Star Formation

Selected with `stellarPopulationSpectraPostprocess<setName>Method=recent` this method suppresses all emission from populations older than `[recentPopulationsTimeLimit]`.

12.34.5. Identity

Selected with `stellarPopulationSpectraPostprocess<setName>Method=identity` this method leaves the spectrum unchanged.

12.35. Stellar Astrophysics

Various properties related to stellar astrophysics are required by GALACTICUS. The following documents their implementation.

12.35.1. 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:

```
<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>
  </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 elementYieldMassX gives the mass of element X 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.

12.35.2. 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.

12.35.3. 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:

stellarTracksFile
| +-> metallicity1
| | +-> metallicity
| | +-> mass1
| | +-> mass
| | +-> age
| | +-> luminosity
| | +-> effectiveTemperature
| | x-> massN
| x-> 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.
12. Physical Implementations

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.

12.35.4. 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.

12.35.5. 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].

12.35.6. 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.

12.36. Substructure and Merging

Substructures and merging of nodes/substructures is controlled by several algorithms which are described below:

12.36.1. Merging Timescales

The method used to compute merging timescales of substructures is specified by the satelliteMergingMethod parameter.
Dynamical Friction: Lacey & Cole

Selected with `satelliteMergingMethod=Lacey-Cole`, 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.

Dynamical Friction: Lacey & Cole + Tormen

Selected with `satelliteMergingMethod=Lacey-Cole+Tormen`, 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:

$$\tau_{\text{merge}} = f_\tau \tau_{\text{dynamical}} \frac{M_{\text{host}}/M_{\text{satellite}}}{\ln(M_{\text{host}}/M_{\text{satellite}})}$$

(12.176)

where $f_\tau = \text{mergingTimescaleMultiplier}$, $\tau_{\text{dynamical}}$ is the dynamical time of the host halo and $B(x) = \text{erf}(x) - 2x \exp(x)/\sqrt{\pi}$. The orbital factor $\Phi = \epsilon^{0.78}(R_c/R_{\text{virial}})^2$ is drawn at random from a log-normal distribution with median $-0.14$ and dispersion $0.26$ as found by Cole et al. [2000].

Dynamical Friction: Jiang (2008)

Selected with `satelliteMergingMethod=Jiang2008`, this method computes merging timescales using the dynamical friction calibration of Jiang et al. [2008].


Selected with `satelliteMergingMethod=BoylanKolchin2008`, this method computes merging timescales using the dynamical friction calibration of Boylan-Kolchin et al. [2008].

Dynamical Friction: Wetzel & White (2010)

Selected with `satelliteMergingMethod=Wetzel-White2010`, this method computes merging timescales using the dynamical friction calibration of Wetzel and White [2010].

12.36.2. 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 `virialOrbitsMethod=Benson2005`, this method selects orbital parameters randomly from the distribution given by Benson [2005].

Fixed

Selected with `virialOrbitsMethod=fixed`, this method sets all orbital parameters to fixed values, with $v_r = [\text{virialOrbitsFixedRadialVelocity}] V_{\text{virial}}$ and $v_\phi = [\text{virialOrbitsFixedTangentialVelocity}] V_{\text{virial}}$. 


12. Physical Implementations

Wetzel (2010)

Selected with \texttt{virialOrbitsMethod=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 $9.999999$\textsuperscript{9}

12.36.3. Node Merging

The algorithm to be used to process nodes when they become substructures is specified by the \texttt{nodeMergersMethod} parameter.

Single Level Hierarchy

Selected with \texttt{nodeMergersMethod=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).

12.37. Supernovae Feedback Models

The supernovae feedback driven outflow rate is computed using the method specified by the \texttt{starFormationFeedback[Disks|Spheroids]Method} for disks and spheroids respectively.

12.37.1. Fixed

Selected with \texttt{starFormationFeedbackDisksMethod=fixed}, this method assumes an outflow rate of:

$$ M_{\text{outflow}} = f_{\text{outflow}} \frac{\dot{E}}{E_{\text{canonical}}}, \quad (12.177) $$

where $f_{\text{outflow}}$ 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 $1M_\odot$ after infinite time.

12.37.2. Power Law

Selected with \texttt{starFormationFeedback[Disks|Spheroids]Method=powerLaw}, this method assumes an outflow rate of:

$$ M_{\text{outflow}} = \left( \frac{V_{\text{outflow}}}{V} \right)^{\alpha_{\text{outflow}}} \frac{\dot{E}}{E_{\text{canonical}}}, \quad (12.178) $$

where $V_{\text{outflow}}$ is the outflow velocity (in km/s) and $\alpha_{\text{outflow}}$ 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.

\textsuperscript{9}We use this value rather than 10 since the GSL $\_2F_1$ hypergeometric function fails in some cases when $C_1 \geq 10$. 
12.37.3. Creasey et al. (2012)

Selected with \texttt{starFormationFeedbackDisksMethod=Creasey2012}, this method computes the outflow rate using the model of Creasey et al. [2012]. Specifically,

\[
M_{\text{outflow}} = \frac{\dot{E}_{\text{SN}}}{E_{\text{SN}} M_*} \int_0^\infty \beta_0 \Sigma_{g,1}^{-1}(r)^f_g(r) \Sigma_*(r) 2\pi r dr,
\]

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, \(\Sigma_*(r)\) is the surface density of star formation rate, \(\dot{M}_*\) is the total star formation rate in the disk, \(\dot{E}_{\text{SN}}\) 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 \(\nu = [\text{starFormationFeedbackDisksCreasy2012Nu}]\).

12.38. Supernovae Expulsive Feedback Models

The expulsive supernovae feedback driven outflow rate is computed using the method specified by the \texttt{starFormationExpulsiveFeedback[Disks|Spheroids]Method} for disks and spheroids respectively.

12.38.1. Null

Selected with \texttt{starFormationExpulsiveFeedback[Disks|Spheroids]Method=null}, this method assumes a zero outflow rate.

12.38.2. Superwind

Selected with \texttt{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.


The method to be used for computing the initial separation of black hole binaries is specified by the \texttt{blackHoleBinaryInitialRadiiMethod} parameter.

12.39.1. Spheroid Radius Fraction

Selected with \texttt{blackHoleBinaryInitialRadiiMethod=spheroidRadiusFraction}, this method assumes that the initial separation of the binary is equal to a fixed fraction \([\text{blackHoleInitialRadiusSpheroidRadiusRatio}]\) of the larger of the spheroid scale radii of the two merging galaxies.
12. Physical Implementations


Selected with `blackHoleBinaryInitialRadiiMethod=Volonteri2003`, this method assumes that the initial separation follows the relationship described in Volonteri et al. [2003]

\[ r_{\text{initial}} = \frac{G(M_{\bullet, 1} + M_{\bullet, 2})}{2\sigma_{\text{DM}}^2} \]  \hspace{1cm} (12.181)

where \( M_{\bullet, 1} \) and \( M_{\bullet, 2} \) are the masses of the black holes and \( \sigma_{\text{DM}} \) is the velocity dispersion of the dark matter, which we assume to equal the virial velocity of the dark matter halo.

12.39.3. 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} (12.182)

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.

12.40. 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.

12.40.1. Null

Selected by default and with `blackHoleBinarySeparationGrowthRateMethod=null`, this method assumes that the initial separation of the binaries is final.

12.40.2. 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.

\[ \dot{a} = \min \left( -\frac{G\rho_* a^2 H}{\sigma}, +\frac{2v_{\text{DF}} a}{v_c} \right) - \frac{256G^3M_{\bullet, 1}M_{\bullet, 2}(M_{\bullet, 1} + M_{\bullet, 2})}{5c^3a^3} \]  \hspace{1cm} (12.183)

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, \( v_{\text{DF}} \) 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:

\[
\dot{v}_{DF} = -\frac{2\pi G^2 M}{V_C^2} \sum_i \rho_i \log(1 + \Lambda_i^2) \left[ \text{erf}(X_i) - \frac{2X_i}{\sqrt{\pi}} \exp\left(-X_i^2\right) \right],
\]

(12.184)

where the sum is taken over the spheroid (gaseous plus stellar mass) and dark matter halo components. Here,

\[
\Lambda_i = \frac{a \sigma_i^2}{G(M_{*,1} + M_{*,2})},
\]

(12.185)

is the Coulomb logarithm and

\[
X_i = \frac{V_c}{\sqrt{2} \sigma}.
\]

(12.186)

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 \(\text{stellarDensityChangeBinaryMotion} = \text{true}\) to account for the ejection of stars from the loss cone.

\[
f_\rho = \min \left\{ \sqrt{\frac{4a \sigma^2_{\text{spheroid}}}{3G(M_{*,1} + M_{*,2})}} \log \left( \frac{GM_{*,2}}{4\sigma^2_{\text{spheroid}}a} \right)^2, 1 \right\},
\]

(12.187)

12.41. 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 \text{blackHoleBinaryRecoilVelocityMethod} parameter.

12.41.1. Null

Selected by default and with \text{blackHoleBinaryRecoilVelocityMethod} = \text{null}, this method assumes that there is no recoil velocity.

12.41.2. Campanelli et al. (2007)

Selected with \text{blackHoleBinaryRecoilVelocityMethod} = \text{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]

\[
V_{\text{recoil}} = V_m \mathbf{e}_1 + V_\perp (\cos \xi \mathbf{e}_1 + \sin \xi \mathbf{e}_2) + V_\parallel \mathbf{e}_2
\]

(12.188)

with:

\[
V_m = A q^2 \left( 1-q \right)^5 \left( 1 + B \frac{q}{(1+q)^2} \right)
\]

(12.189)

\[
V_\perp = H \frac{q^2}{(1+q)^5} \left( \alpha_2 - q \alpha_1 \right)
\]

(12.190)

[10] The disk is ignored as the black hole is assumed to be orbitting in a circular orbit in the disk.
\[ V_\parallel = K \cos(\theta - \theta_0) \frac{q^2}{(1 + q)^5} (\alpha_2^\perp - q\alpha_1^\perp) \]  

(12.191)

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. \( \hat{e}_1, \hat{e}_2 \) 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 \( H = (7.3 \pm 0.3) \times 10^3 \text{ km/s} \), González et al. [2007b] for \( A = 1.2 \times 10^4 \text{ km/s} \) \( B = -0.93 \), González et al. [2007a] for \( K \cos(\delta \theta) = (6.0 \pm 0.1) \times 10^4 \text{ km/s} \) and \( K = (6.0 \pm 0.1) \times 10^4 \text{ km/s} \).

12.42. 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 \texttt{blackHoleBinaryMergersMethod} parameter.

12.42.1. Rezzolla et al. (2008)

Selected with \texttt{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. Additional Output Quantities

13.1. 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 $M_\odot$ Gyr$^{-1}$;
- `blackHoleJetPower` The power being emitted into jets by the black hole/accretion disk system in $M_\odot$ km$^2$ s$^{-2}$ Gyr$^{-1}$.

13.2. 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 $M_\odot$ Gyr$^{-1}$;
- `hotHaloCoolingRadius` The characteristic cooling radius in the halo in Mpc.

13.3. 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.

13.4. 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).

13.5. 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 \(M_\odot\)).

13.6. 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 nodeIndex for non-satellite nodes.

In addition to these quantities output for each node, setting \([\text{outputHaloModelData}] = \text{true}\) causes the creation of a haloModel group in the Galacticus output file. This group contains the following:

wavenumber A dataset giving the wavenumbers (in units of 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 Mpc\(^3\) normalized to \(z = 0\)) at each wavenumber specified in the wavenumber dataset.

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 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).

13.7. 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 §16.4.3). In such cases, the following properties will be output for all galaxies:
13.7. Lightcone Coordinates

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 lightcone1;

angularWeight The mean number density of this galaxy per unit area on the sky (in degrees^-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:

```xml
<geometry>
  <boxLength>500</boxLength>
  <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>
  </outputs>
</geometry>
```

1Note that this will not, in general, be precisely the same as the redshift corresponding to the output time.
13. Additional Output Quantities

...<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>
<unitVector2>
  <coordinate>0</coordinate>
  <coordinate>0.923076923076923</coordinate>
  <coordinate>-0.230769230769231</coordinate>
</unitVector2>
<unitVector3>
  <coordinate>-0.905325443786982</coordinate>
  <coordinate>0.0710059171597633</coordinate>
  <coordinate>0.284023668639053</coordinate>
</unitVector3>
<units>
  <length>
    <hubbleExponent>-1</hubbleExponent>
    <unitsInSI>3.08568025e+22</unitsInSI>
  </length>
</units>
</geometry>

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.
13.8. 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 \text{mainProgenitorEvolution} in the GALACTICUS output file. Within that group two datasets, \text{time} and \text{expansionFactor}, give the times and expansion factors at which evolution was recorded. Then for each merger tree two datasets, \text{stellarMass}<N> and \text{totalMass}<N> (where \text{<N>} is the merger tree index), give the stellar and total baryonic mass of the main branch progenitor at each timestep.

13.9. 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.

13.10. 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, \text{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}.

13.11. Merger Tree Links and Node Isolation

The following properties are output to permit the merger tree structure to be recovered:
\text{nodeIndex} A unique (within a tree) integer index identifying the node;
\text{childIndex} The index of this node’s primary child node (or \(-1\) if it has no child);
\text{parentIndex} The index of this node’s parent node (or \(-1\) if it has no parent);
\text{SiblingIndex} The index of this node’s sibling node (or \(-1\) if it has no sibling);
\text{satelliteIndex} The index of this node’s first satellite node (or \(-1\) if it has no satellites);
\text{nodeIsIsolated} Will be \text{0} for a node which is a subhalo inside some other node (i.e. a satellite galaxy) or \text{1} for a node that is an isolated halo (i.e. a central galaxy).

The \text{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 [nodePromotionIndexShift]=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 13.1.

13.12. 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 render_⟨treeIndex⟩_- ⟨outputIndex⟩.hdf5 where ⟨treeIndex⟩ is the index of the tree and ⟨outputIndex⟩ 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;
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.

13.13. 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 [mergerTreesDumpStructure]=true. After each evolution step the tree structure will be dumped to a file named mergerTreeDump:⟨treeIndex⟩:⟨outputIndex⟩.gv where ⟨treeIndex⟩ is the index of the tree and ⟨outputIndex⟩ 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.


Setting [outputMostMassiveProgenitor]=true causes the property 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.

13.15. Rotation Curve

Setting [outputRotationCurveData]=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 outputRotationCurveRadii parameter. This parameter’s value can contain multiple entries. Each entry is of the form:
Figure 13.1.: 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.
13. Additional Output Quantities

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.

13.16. 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.

13.17. 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.

13.18. 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 blackHole;

massType specifies which type of mass the velocity dispersion should be computed for—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 velocity dispersion;

direction should be one of radial (computes the radial component of velocity dispersion), lineOfSight{<luminosity>} (computes the line-of-sight velocity dispersion), or lineOfSightInteriorAverage{<luminosity>} (computes the line-of-sight velocity dispersion averaged interior to the given radius)—in the latter two 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.

13.19. 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).

2Note that attempting to compute the velocity dispersion for a black hole or disk for example won’t make any sense.
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.
14. 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**.

14.1. Getting Started

It’s easy to begin working with and changing the GALACTICUS source code. Assuming you have Mercurial ("hg") installed, just do:

```
hg clone https://abensonca@bitbucket.org/abensonca/galacticus galacticus
```

and you have a cloned copy of the GALACTICUS repository in the `galacticus` directory.

14.1.1. 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.

14.2. 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**:

```
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:
14. Developing GALACTICUS

hg incoming

If any new changesets are shown, use `hg pull -u` to merge these into your working copy. Then commit your changes:

```bash
hg commit
```

Your changes are now committed to your cloned repository.

14.2.1. 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:

```bash
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 attach the `changes.diff` file to an e-mail. It will be merged into the GALACTICUS project using

```bash
hg import changes.diff
```

**Using BitBucket**

If you forked the GALACTICUS repository on BitBucket, you can now push your changes back to BitBucket using

```bash
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.

14.3. 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:

```bash
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`):
14.4. Releases

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).

14.4. Releases

Each release of Galacticus exists as a separate branch within the main Galacticus repository\(^1\). 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

14.4.1. 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.

---

\(^1\)At least they will, beginning from the v0.9.2 release.
15. Coding **GALACTICUS**

15.1. 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.

15.1.1. 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 &
& )
```

253
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.
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 CMBFast transfer function method (see §12.4.5), which computes a transfer function using CMBFAST 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 CMBFast transfer function method the following code is used:

```xml
!# <uniqueLabel>
!# <function>Transfer_Function_CMBFast_Label</function>
!# <ignore>transferFunctionFile</ignore>
!# <ignoreRegex>^imf.*</ignoreRegex>
!# </uniqueLabel>
```

This XML block specifies that the labelling function should be called `Transfer_Function_CMBFast_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 CMBFast 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.

The function returns a `type(varying_string)` object with the encoded label. For the CMBFast method this might be:

```fortran
write (0,*) char(Transfer_Function_CMBFast_Label())
::cosmological_parameters#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_CMBFast_Label(includeVersion=.true.))
::cosmological_parameters#Omega_b[0.0455]#Omega_Matter[0.2725]#Omega_DE[0.7275]#T_CMB[2.72548]#H_0[70.2]_v0.9.1.r764
```

This can be useful to ensure that a file was generated by the exact same version of GALACTICUS. Finally, 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:

```fortran
write (0,*) char(Transfer_Function_CMBFast_Label(includeVersion=.true.,asHash=.true.))
tudE3GwpjAXBzppr5azU0
```
This hash can be used as part of the name of the file to which data is written.

15.3. 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.

15.3.1. 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\(^1\). 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 §16.4.3). 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.

15.4. 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 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.

15.4.1. 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>();
```

\(^1\)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.
15.4.2. 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**

```
use Cosmology_Functions
expansionFactor=Expansion_Factor(cosmologicalTime)
```

**C++**

```cpp
#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.

15.4.3. 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
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.

15.5. Objects

15.5.1. Enumerations

Enumerations are used to communicate options to many functions in GALACTICUS. All available enumerations, along with their members, are described below.
adjustElements

Description: Used to specify how elements should be adjusted when the metallicity of an abundances object is changed.

Provided by: module Abundances_Structure

Members:
- adjustElementsNone
- adjustElementsReset
- adjustElementsUpdate

componentType

Description: Used to specify the component(s) to be queried in galactic structure functions.

Provided by: module Galactic_Structure_Options

Members:
- componentTypeUnknown
- componentTypeAll
- componentTypeDisk
- componentTypeSpheroid
- componentTypeHotHalo
- componentTypeDarkHalo
- componentTypeBlackHole

coordinateSystem

Description: Used to specify the coordinate system of the input coordinates in galactic structure functions.

Provided by: module Galactic_Structure_Options

Members:
- coordinateSystemSpherical
- coordinateSystemCylindrical
- coordinateSystemCartesian

dataType

Description: Used to specify the type of data being stored in a mergerTreeData structure metadata entry.

Provided by: module Merger_Tree_Data_Structure

Members:
- dataTypeInteger
- dataTypeDouble
- dataTypeText

extrapolationType

Description: Used to specify the type of extrapolation to use when interpolating in tables.

Provided by: module Tables

Members:
- extrapolationTypeExtrapolate
- extrapolationTypeFix
- extrapolationTypeAbort
massDistributionSymmetry

Description: Used to specify the symmetry of massDistribution objects.
Provided by: module Mass_Distributions
Members: massDistributionSymmetryNone
massDistributionSymmetryCylindrical
massDistributionSymmetrySpherical

massType

Description: Used to specify the mass type(s) to be queried in galactic structure functions.
Provided by: module Galactic_Structure_Options
Members: massTypeUnknown
massTypeAll
massTypeDark
massTypeBaryonic
massTypeGalactic
massTypeGaseeous
massTypeStellar
massTypeBlackHole

metaDataType

Description: Used to specify the type of metadata being stored in a mergerTreeData structure.
Provided by: module Merger_Tree_Data_Structure
Members: metaDataGeneric
metaDataCosmology
metaDataSimulation
metaDataGroupFinder
metaDataTreeBuilder
metaDataProvenance

metallicityScale

Description: Used to specify the metallicity scale when working with abundances objects.
Provided by: module Abundances_Structure
Members: linearByMass
linearByNumber
logarithmicByMassSolar
logarithmicByNumberSolar
linearByMassSolar
linearByNumberSolar
propertyType

Description: Used to specify properties in a mergerTreeData structure.
Provided by: module Merger_Tree_Data_Structure
Members:
- propertyTypeTreeIndex
- propertyTypeNodeIndex
- propertyTypeDescendentIndex
- propertyTypeHostIndex
- propertyTypeRedshift
- propertyTypeNodeMass
- propertyTypeParticleCount
- propertyTypePositionX
- propertyTypePositionY
- propertyTypePositionZ
- propertyTypeVelocityX
- propertyTypeVelocityY
- propertyTypeVelocityZ
- propertyTypeSpinX
- propertyTypeSpinY
- propertyTypeSpinZ
- propertyTypeSpin
- propertyTypeAngularMomentumX
- propertyTypeAngularMomentumY
- propertyTypeAngularMomentumZ
- propertyTypeAngularMomentum
- propertyTypeHalfMassRadius
- propertyTypeParticleIndex
- propertyTypeMostBoundParticleIndex
- propertyTypeSnapshot

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: module Root_Finder
Members:
- rangeExpandNull
- rangeExpandAdditive
- rangeExpandMultiplicative

rangeExpandSignExpect

Description: Used to specify the expected sign of the root function when searching for roots using a rootFinder object.
Provided by: module Root_Finder
Members:
- rangeExpandSignExpectNegative
- rangeExpandSignExpectNone
- rangeExpandSignExpectPositive
15.5. Objects

units
Description: Used to specify the type of units being stored in a mergerTreeData structure.
Provided by: module Merger_Tree_Data_Structure
Members:
  - unitsMass
  - unitsLength
  - unitsTime
  - unitsVelocity

weightBy
Description: Used to specify by which quantity to weight the results in galactic structure functions.
Provided by: module Galactic_Structure_Options
Members:
  - weightByMass
  - weightByLuminosity

15.5.2. 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).

abundances
  - <type(abundances)> add(<type(abundances)> abundances2) Add two abundances.
  - <void> builder(<*type(node)> abundancesDefinition) Build an abundances object from a provided XML description.
  - <void> deserialize(<double(:)> historyArray) Deserialize an abundances object from an array.
  - <void> destroy() Destroy an abundances object.
  - <type(abundances)> divide(<double> divisor) Divide an abundance by a scalar.
  - <void> dump() Dump an abundances object.
  - <void> dumpRaw(<integer> fileHandle) Dump an abundances object to binary.
  - <double> heliumMassFraction() Returns the helium fraction by mass.
  - <double> heliumNumberFraction() Returns the helium fraction by number.
  - <double> hydrogenMassFraction() Returns the hydrogen fraction by mass.
  - <double> hydrogenNumberFraction() Returns the hydrogen fraction by number.
  - <void> increment(<type(abundances)> addAbundances) Increment an abundances object.
  - <logical> isZero() Return true if an abundances object is zero.
  - <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.
  - <double> metallicity(<metallicityScale> [metallicityType]) Returns the metallicity.
15. Coding GALACTICUS

```cpp
<metalicitySet<double> metallicity>v, <metallicityScale> [metallicityType]v, <adjustElements> [adjustElements]v, <integer> [abundanceIndex]v) Sets the metallicity to metallicity.

<type(abundances)> multiply<double> multiplier>v Multiply an abundance by a scalar.

<void> output<integer> integerPropertyv, <integer> integerBufferCountv, <integer(,:)> integerBufferv, <integer> doublePropertyv, <integer> doubleBufferCountv, <double(,:)> doubleBufferv, <double> timev, <integer> instancev) Store an abundances object in the output buffers.

<void> outputCount<integer> integerPropertyCountv, <integer> doublePropertyCountv, <double> timev, <integer> instancev) Specify the count of an abundances object for output.

<void> outputNames<integer> integerPropertyv, <char[*]> integerPropertyNamesv, <char[*]> integerPropertyCommentsv, <double(,:)> integerPropertyUnitsSIV, <integer> doublePropertyv, <char[*]> doublePropertyNamesv, <char[*]> doublePropertyCommentsv, <double(,:)> doublePropertyUnitsSIV, <double> timev, <integer> instancev) Specify the names of abundance object properties for output.

<void> readRaw<integer> fileHandlev) Read an abundances object from binary.

<void> reset() Reset an abundances object.

<void> serialize<double(,:)> historyArrayv) Serialize an abundances object to an array.

<integer> serializeCount() Return a count of the number of properties in a serialized abundances object.

<void> setToUnity() Set an abundances object to unity.

<type(abundances)> subtract(type(abundances)> abundances2) Subtract one abundance from another.

chemicalAbundances

<double> abundance<integer> moleculeIndexv) Returns the abundance of a chemical given its index.

<void> abundanceSet<integer> moleculeIndexv, <double> abundancev) Sets the abundance of a chemical given its index.

<type(chemicalAbundances)> add(type(chemicalAbundances)> abundances2) Add two chemical abundances.

<void> builder(type(node)> chemicalAbundancesDefinitionv) Build a chemical abundances object from an XML definition.

<void> deserialize<double(,:)> chemicalAbundancesArrayv) Deserialize a chemical abundances object from an array.

<void> destroy() Destroys a chemical abundances object.

<type(chemicalAbundances)> divide<double> divisorv) Divide a chemical abundance by a scalar.

<void> dump() Dump a chemical abundances object.
```
15.5. Objects

<void> dumpRaw(<integer> fileHandle) Dump a chemical abundances object in binary.

<void> enforcePositive() Enforces all chemical values to be positive.

<void> increment(<type(chemicalAbundances)> addAbundances) Increment a chemical abundances object.

<void> massToNumber(<type(chemicalAbundances)> chemicalsByNumber) Converts from abundances by mass to abundances by number.

<type(chemicalAbundances)> multiply(<double> multiplier) Multiply a chemical abundance by a scalar.

<void> numberToMass(<type(chemicalAbundances)> chemicalsByMass) Converts from abundances by number to abundances by mass.

<void> readRaw(<integer> fileHandle) Read a chemical abundances object in binary.

<void> reset() Resets abundances to zero.

<void> serialize(<double(:)> chemicalAbundancesArray) Serialize a chemical abundances object to an array.

<integer> serializeCount() Return a count of the number of properties in a serialized chemical abundances object.

<void> setToUnity() Set abundances to unity.

<type(chemicalAbundances)> subtract(<type(chemicalAbundances)> abundances2) Subtract one chemical abundance from another.

chemicalStructure

<integer> charge() Return the charge of a chemical.

<void> export(<character(len=*>) coutputFile) Write a chemical structure to a CML file.

<double> mass() Return the mass of a chemical in atomic mass units.

<void> retrieve(<character(len=*>) chemicalName) Get a chemical from the database.

coordinate

<void> fromCartesian(<double(3)> x) Set the coordinates from a Cartesian system specified as a 3-element array.

<double(3)> toCartesian() Return the coordinates in a Cartesian system as a 3-element array.
15. Coding Galacticus

coordinateCartesian

<void> fromCartesian(<double(3)> x) Set the coordinates from a Cartesian system specified as a 3-element array.

<double(3)> toCartesian() Return the coordinates in a Cartesian system as a 3-element array.

<double> x() Get the x-coordinate.

<void> xSet(<double> x) set the x-coordinate.

<double> y() Get the y-coordinate.

<void> ySet(<double> y) set the y-coordinate.

<double> z() Get the z-coordinate.

<void> zSet(<double> z) set the z-coordinate.

coordinateCylindrical

<void> fromCartesian(<double(3)> x) Set the coordinates from a Cartesian system specified as a 3-element array.

<double> phi() Get the φ-coordinate.

<void> phiSet(<double> phi) set the φ-coordinate.

<double> r() Get the r-coordinate.

<void> rSet(<double> r) set the r-coordinate.

<double(3)> toCartesian() Return the coordinates in a Cartesian system as a 3-element array.

<double> z() Get the z-coordinate.

<void> zSet(<double> z) set the z-coordinate.

coordinateSpherical

<void> fromCartesian(<double(3)> x) Set the coordinates from a Cartesian system specified as a 3-element array.

<double> phi() Get the φ-coordinate.

<void> phiSet(<double> phi) set the φ-coordinate.

<double> r() Get the r-coordinate.

<void> rSet(<double> r) set the r-coordinate.

<double> theta() Get the θ-coordinate.

<void> thetaSet(<double> theta) set the θ-coordinate.

<double(3)> toCartesian() Return the coordinates in a Cartesian system as a 3-element array.
15.5. Objects

hashPerfect

<void> create(<integer(kind=kind_int8)(::)> keys →, <integer(kind=kind_int8)(::)>[values] →, <logical>[keepInverseTable]→) Create a perfect hash.

<void> destroy() Destroy a perfect hash.

<integer(kind=kind_int8)> index(<integer(kind=kind_int8)> key →) Return the index corresponding to a key in a perfect hash.

<logical> isPresent(<integer(kind=kind_int8)> key →) Test if a key is present in a perfect hash.

<integer(kind=kind_int8)> size() Return the size of a perfect hash.

<integer(kind=kind_int8)> value(<integer(kind=kind_int8)> key →) Return the value corresponding to a key in a perfect hash.

hdf5Object

<void> assertAttributeType(<integer(kind=HID_T)(::)> attributeAssertedType →, <integer>attributeAssertedRank→, [matches]←) Check the type and rank of an attribute.

<void> assertDatasetType(<integer(kind=HID_T)(::)> datasetAssertedType →, <integer> datasetAssertedRank→) Check the type and rank of a dataset.

<void> close() Close an HDF5 object.

<void> createReference1D(<type(hdf5Object)> toDataset ↔, <character(len=*)> referenceName→, <integer(kind=HSIZE_T)(1)> referenceStart→, <integer(kind=HSIZE_T)(1)> referenceCount→) Create a reference to a 1D dataset.

<void> createReference2D(<type(hdf5Object)> toDataset ↔, <character(len=*)> referenceName→, <integer(kind=HSIZE_T)(2)> referenceStart→, <integer(kind=HSIZE_T)(2)> referenceCount→) Create a reference to a 2D dataset.

<void> createReference3D(<type(hdf5Object)> toDataset ↔, <character(len=*)> referenceName→, <integer(kind=HSIZE_T)(3)> referenceStart→, <integer(kind=HSIZE_T)(3)> referenceCount→) Create a reference to a 3D dataset.

<void> createReference4D(<type(hdf5Object)> toDataset ↔, <character(len=*)> referenceName→, <integer(kind=HSIZE_T)(4)> referenceStart→, <integer(kind=HSIZE_T)(4)> referenceCount→) Create a reference to a 4D dataset.

<void> createReference5D(<type(hdf5Object)> toDataset ↔, <character(len=*)> referenceName→, <integer(kind=HSIZE_T)(5)> referenceStart→, <integer(kind=HSIZE_T)(5)> referenceCount→) Create a reference to a 5D dataset.

<void> destroy() Destroy an HDF5 object.

<logical> hasAttribute(<character(len=*)> [attributeName]→) Check if an object has a named attribute.

<logical> hasDataset(<character(len=*)> [datasetName]→) Check if an object has a named dataset.

<logical> hasGroup(<character(len=*)> [groupName]→) Check if an object has a named group.
15. Coding Galacticus

<logical> isOpen() Return true if an object is open.

<logical> isReference() Return true if a dataset is a reference.

<type(hdf5Object openAttribute(<character(len=*)) attributeName →, <integer> [attributeDataType] →, <integer(kind=HSIZE_T)():] [attributeDimensions] →, <logical> [isOverwritable] →, <integer(kind=HID_T)> [useDataType] →) Open an HDF5 attribute.

<type(hdf5Object openDataset(<character(len=*)) datasetName →, <character(len=*)> [commentText] →, <integer> [datasetDataType] →, <integer(kind=HSIZE_T)():] [datasetDimensions] →, <logical> [isOverwritable] →, <logical> [appendTo] →, <integer(kind=HID_T)> [useDataType] →, <integer> [chunkSize] →, <integer> [compressionLevel] →) Open an HDF5 dataset.


<type(hdf5Object openGroup(<character(len=*)) groupName →, <character(len=*)> [commentText] →, <logical> [overWrite] →, <logical> [objectsOverwritable] →, <integer> [chunkSize] →, <integer> [compressionLevel] →) Open an HDF5 group and return an appropriate HDF5 object.

<type(varying_string)> pathTo() Returns the path to a given object.

<void> readAttribute(<character(len=*)) [attributeName] →, <(integer|integer(kind=int8)|double|character(len=*)|type(varying_string))[0-1]*] > attributeValue →, <logical> [allowPseudoScalar] →) Read an attribute from an HDF5 object.

<void> readAttributeStatic(<character(len=*)) [attributeName] →, <(integer|integer(kind=int8)|double|character(len=*)|type(varying_string))[0-1]*] > attributeValue →, <logical> [allowPseudoScalar] →) Read an attribute from an HDF5 object into a static array.

<void> readDataset(<character(len=*)) [datasetName] →, <(integer|integer(kind=int8)|double|character(len=*)|type(varying_string))[0-1]*] > datasetValue →, <integer(kind=HSIZE_T)[1]> [readBegin] →, <integer(kind=HSIZE_T)[1]> [readCount] →) Read a dataset from an HDF5 group into an allocatable array.

<void> readDatasetStatic(<character(len=*)) [datasetName] →, <(integer|integer(kind=int8)|double|character(len=*)|type(varying_string))[0-1]*] > datasetValue →, <integer(kind=HSIZE_T)[1]> [readBegin] →, <integer(kind=HSIZE_T)[1]> [readCount] →) Read a dataset from an HDF5 group into a static array.

<integer(kind=HSIZE_T)> size(<integer> [dim] →) Return the size of a dataset.

<void> writeAttribute(<(integer|integer(kind=int8)|double|character(len=*)|type(varying_string))[0-1]*] > attributeValue →, <character(len=*)> [attributeName] →)) Write an attribute to an HDF5 object.

<void> writeDataset(<(integer|integer(kind=int8)|double|character(len=*)|type(varying_string))[0-1]*] > datasetValue →, <character(len=*)> [datasetName] →, <character(len=*)> [commentText] →, <logical> [appendTo] →, <integer> [chunkSize] →, <integer> [compressionLevel] →, <type(hdf5Object) [datasetReturned] →) Write a dataset to an HDF5 group.

\[\text{For double datasets, up to 5-dimensional datasets are supported.}\]
15.5. Objects

history

<type(history)> add(<type(history)> + <type(history)>) Addition operator.

<void> builder(<type(node)> historyDefinition) Build a history object from an XML definition.

<void> clone(<type(history)> historyToClone) Clone a history object.

<void> combine(<type(history)> combineHistory) Combines two histories.

<void> create(<integer> historyCount, <integer> timesCount, <double> [timeBegin], <double> [timeEnd], <integer> [rangeType]) Creates a history object with a specified range of times.

<void> deserialize(<double[:]) historyArray) Deserializes a history object from an array.

<void> destroy(<type(history)> historyToClone) Destroys a history object.

<type(history)> divide(<type(history)> / <type(history)>) Division operator.

<void> dump() Dump a history object.

<void> dumpRaw(<integer> fileHandle) Dump a history object in binary.

<logical> exists() Returns true if the given history has been created.

<void> extend(<double(2) [timeRange], <double[:]) [times]) Extends the time range of a history to encompass the specified limits.

<void> increment(<type(history)> addHistory) Adds two histories, possibly with different time series.

<logical> isZero() Returns true if the history is entirely zero.

<void> readRaw(<integer> fileHandle) Read a history object in binary.

<void> reset() Resets all entries in a history to zero.

<void> serialize(<double[:]) historyArray) Serializes a history object to an array.

<integer> serializeCount() Return a count of the number of properties in a serialized history object.

<void> setToUnity() Set all entries in a history to unity.

<type(history)> subtract(<type(history)> - <type(history)>) Subtraction operator.

<void> timeSteps(<double[:]) timeSteps) Returns an array with the timesteps (i.e. the intervals between successive times) in the given history.

<void> trim(<double> currentTime, <integer> [minimumPointsToRemove]) Removes any times in a history which have become outdated.
integerScalarHash

<void> delete(<(character(len=*)|varying_string)> key→, <integer> value→) Delete a key from the hash.

<logical> exists(<(character(len=*)|varying_string|<integer>>) key→) Return true if the specified key exists in the hash.

<void> initialize() Initialize the hash.

<type(varying_string)> key(<integer> indexValue→) Return the key of the indexValue\textsuperscript{th} entry in the hash.

<void> keys(<type(varying_string)[::]> keys↔) Return an array of all keys in the hash.

<void> set(<(character(len=*)|varying_string)> key→, <integer> value→) Set the value of a key in the hash.

<integer> size() Return the number of keys in the hash.

<integer> value(<(character(len=*)|varying_string|<integer>>) key→, <integer> value→) Return the value for the given key.

<void> values(<integer[::]> values↔) Return an array of all values in the hash.

keplerOrbit

<double> angularMomentum() Returns the angular momentum of an orbit.

<void> angularMomentumSet(<double> angularMomentum→) Sets the angular momentum of an orbit.

<void> assertIsDefined() Asserts that an orbit is fully defined.

<void> builder(<*type(node)> keplerOrbitDefinition→) Build a Kepler orbit from an XML definition.

<void> destroy() Destroys an orbit.

<void> dump() Dump an orbit.

<void> dumpRaw(<integer> fileHandle→) Dump an orbit in binary.

<double> eccentricity() Returns the eccentricity of an orbit.

<void> eccentricitySet(<double> eccentricity→) Sets the eccentricity of an orbit.

<double> energy() Returns the energy of an orbit.

<void> energySet(<double> energy→) Sets the energy of an orbit.

<double> hostMass() Returns the host mass of an orbit.

<logical> isBound() Returns true if the orbit is bound.

<logical> isDefined() Returns true if an orbit is fully defined.

<void> massesSet(<double> satelliteMass→, <double> hostMass→) Sets the masses of satellite and host objects.
15.5. Objects

<void> output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer> integerBuffer↔, <integer> doubleProperty↔, <integer> doubleBufferCount↔, <double> doubleBuffer↔, <double> time→, <integer> instance→) Store a keplerOrbit object in the output buffers.

<void> outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Specify the count of a keplerOrbit object for output.

<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 a keplerOrbit object properties for output.

<void> propagate(<double> velocityRadial→) Propagates an orbit to a new position.

<double> radius() Returns the radius of an orbit.

<double> radiusApocenter() Returns the apocenter radius of an orbit.

<void> radiusApocenterSet(<double> radius→) Sets the apocenter radius of an orbit.

<double> radiusPericenter() Returns the pericenter radius of an orbit.

<void> radiusPericenterSet(<double> radius→) Sets the pericenter radius of an orbit.

<double> radiusSet(<double> radius→) Sets the radius of an orbit.

<void> readRaw(<integer> fileHandle→) Read an orbit in binary.

<void> reset() Resets an orbit to a null state.

<double> semiMajorAxis() Returns the semi-major axis of an orbit.

<void> semiMajorAxisSet(<double> semiMajorAxis→) Sets the semi-major axis of an orbit.

<double> specificReducedMass() Returns the specific reduced mass (i.e. the reduced mass per unit satellite mass, μ_s = M_{host}/(M_{satellite} + M_{host})) of the orbit.

<double> velocityRadial() Returns the radial velocity of an orbit.

<void> velocityRadialSet(<double> newRadius→) Sets the radial velocity of an orbit.

<double> velocityScale() Returns the velocity scale of an orbit.

<double> velocityTangential() Returns the tangential velocity of an orbit.

<void> velocityTangentialSet(<double> velocityTangential→) Sets the tangential velocity of an orbit.
massDistribution

<double> density(<class(coordinate)> coordinates→) Returns the density of the mass distribution at the supplied coordinates.

<double> densityRadialMoment(<double> moment→, <logical> [isInfinite]←) Returns the \( n^{\text{th}} \) moment of the integral of the density over radius, \( \int_0^\infty \rho(x)|x|^n \, dx \).

<logical> isDimensionless() Returns true if this is a dimensionless mass distribution, false otherwise.

<double> massEnclosedBySphere(<double> radius→) Returns the mass enclosed by a sphere of given radius centered on the origin.

<double> potential(<class(coordinate)> coordinates→) Returns the gravitational potential at the specified coordinates.

<massDistributionSymmetry> symmetry() Returns a label specifying the symmetry of the mass distribution (see §16.5.4).

massDistributionCylindrical

<double> density(<class(coordinate)> coordinates→) Returns the density of the mass distribution at the supplied coordinates.

<double> densityRadialMoment(<double> moment→, <logical> [isInfinite]←) Returns the \( n^{\text{th}} \) moment of the integral of the density over radius, \( \int_0^\infty \rho(x)|x|^n \, dx \).

<logical> isDimensionless() Returns true if this is a dimensionless mass distribution, false otherwise.

<double> massEnclosedBySphere(<double> radius→) Returns the mass enclosed by a sphere of given radius centered on the origin.

<double> potential(<class(coordinate)> coordinates→) Returns the gravitational potential at the specified coordinates.

<massDistributionSymmetry> symmetry() Returns a label specifying the symmetry of the mass distribution (see §16.5.4).

massDistributionSpherical

<double> density(<class(coordinate)> coordinates→) Returns the density of the mass distribution at the supplied coordinates.

<double> densityRadialMoment(<double> moment→, <logical> [isInfinite]←) Returns the \( n^{\text{th}} \) moment of the integral of the density over radius, \( \int_0^\infty \rho(x)|x|^n \, dx \).

<double> halfMassRadius() Returns the radius enclosing half of the mass of the mass distribution.

<logical> isDimensionless() Returns true if this is a dimensionless mass distribution, false otherwise.

<double> massEnclosedBySphere(<double> radius→) Returns the mass enclosed by a sphere of given radius centered on the origin.
15.5. Objects

<double> potential(<class(coordinate)> coordinates→) Returns the gravitational potential at the specified coordinates.

<massDistributionSymmetry> symmetry() Returns a label specifying the symmetry of the mass distribution (see §16.5.4).

mergerTreeData

<void> addMetadata(<metaDataType> metadataType→, <character(len=*)> label→, ((<double>,<integer>,character(len=*)),value→)) Add a metadatum to the tree data structure.

<void> export(<character(len=*)> outputFileName→, <character(len=*)> outputFormat→, <integer> hdfChunkSize→, <integer> hdfCompressionLevel→, <logical> [append]→) Export the tree data to an output file.

<void> makeReferences(<logical> makeReferences→) Specify whether or not merger tree dataset references should be made.

<void> nodeCountSet(<integer> nodeCount→) Set the total number of nodes in the data structure.

<void> particleCountSet(<integer> particleCount→) Set the total number of particles in the data structure.

<void> readASCII(inputFile→, [lineNumberStart]→, [lineNumberStop]→, [separator]→) Read node data from an ASCII file into the data structure.

<void> readParticlesASCII(<character(len=*>) inputFile→, <integer> [lineNumberStart]→, <integer> [lineNumberStop]→, <character(len=*>) [separator]→) Read particle data from an ASCII file into the data structure.

<void> reset() Reset the data structure.

<void> setIncludesHubbleFlow(<logical> includesHubbleFlow→) Specify if velocities include the Hubble flow.

<void> setIncludesSubhaloMasses(<logical> includesSubhaloMasses→) Set whether halo masses include the masses of the subhalos.

<void> setParticleMass(<double> particleMass→) Set the mass of an N-body particle in the simulation from which the trees were derived.

<void> setParticlePropertyColumn(<propertyType> propertyType→, <integer(kind=kind_int8)(::)|<double(::)> property→) Set a node property in the data structure.

<void> setPositionsArePeriodic(<logical> isPeriodic→) Set if positions are periodic.

<void> setProperty(<propertyType> propertyType→, <integer(kind=kind_int8)(::)|<double(::)> property→) Set a node property in the data structure.

<void> setPropertyColumn(<propertyType> propertyType→, columnName→) Set the column in an ASCII data file corresponding to a given node property.

<void> setSelfContained(<logical> areSelfContained→) Specify if trees are self-contained (i.e. contain no cross-links to other trees).
15. Coding GALACTICUS

```c
<void> setUnits(<metaDataType> unitType →, <double> unitsInSI →, <integer> [hubbleExponent] →, <integer> [scaleFactorExponent] →, <character(len=*>> [name] →) Set the units used.

<void> treeCountSet(<integer> treeCount →) Set the total number of trees in the data structure.

nodeComponent

<double> 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 ←) Deserializetheevolvablequantitiesfromanarray.

<void> destroy() Destroy the object.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle →) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the mass enclosed within a radius.

<type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances) value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> nameFromIndex(<integer> count →, <varying_string>name ←) Return the name of a property given is index.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty ←, <integer> integerBufferCount ←, <integer(:, :) integerBuffer ←, <integer> doubleProperty←, <integer> doubleBufferCount ←, <double(:, :) doubleBuffer ←, <double> time →, <integer> instance →) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount ←, <integer> doublePropertyCount ←, <double> time →, <integer> instance →) Compute a count of outputtable properties.
```
15.5. Objects

```c
<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 output-puttable properties.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double::> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double::> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double::> array→) Serialize the evolvable quantities to an array.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

nodeComponentBasic

<double> accretionRate() Get the accretionRate property of the basic component.

<double> accretionRateIsGettable() Get the accretionRate property of the basic component.

<logical> accretionRateIsSettable() Specify whether the accretionRate property of the basic component is settable.

<void> accretionRateSet(<double> value) Set the accretionRate property of the basic component.

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.
```
\[\text{void} \ \text{dumpRaw}(<\text{integer}\ \text{fileHandle}>)\ \text{Generate a binary dump of all properties.}\]

\[<\text{double}> \ \text{enclosedMass}(<\text{double}\ \text{radius}>, <\text{componentType}\ [\text{componentType}]>, <\text{massType}\ [\text{massType}]>, <\text{weightBy}\ [\text{weightBy}]>, <\text{integer}\ [\text{weightIndex}]>, <\text{logical}\ [\text{haloLoaded}]>)\ \text{Compute the mass enclosed within a radius.}\]

\[<\text{type(treeNode)}> \ \text{host}()\ \text{Return a pointer to the host treeNode object.}\]

\[<\text{void}> \ \text{hotHaloCoolingAbundancesRate}(<\text{type(abundances)}\ \text{value})\ \text{Cumulate to the rate of the hotHalo}{\text{coolingAbundances}}\ \text{property of the hotHalo component.}\]

\[<\text{void}> \ \text{hotHaloCoolingAngularMomentumRate}(<\text{double}\ \text{value})\ \text{Cumulate to the rate of the hotHalo}{\text{coolingAngularMomentum}}\ \text{property of the hotHalo component.}\]

\[<\text{void}> \ \text{hotHaloCoolingMassRate}(<\text{double}\ \text{value})\ \text{Cumulate to the rate of the hotHalo}{\text{coolingMass}}\ \text{property of the hotHalo component.}\]

\[<\text{void}> \ \text{initialize}()\ \text{Initialize the object.}\]

\[<\text{double}> \ \text{mass}()\ \text{Get the mass property of the basic component.}\]

\[<\text{integer}> \ \text{massCount}()\ \text{Compute the count of evolvable quantities in the mass property of the BasicStandard component.}\]

\[<\text{double}> \ \text{massIsGettable}()\ \text{Get the mass property of the basic component.}\]

\[<\text{logical}> \ \text{massIsSettable}()\ \text{Specify whether the mass property of the basic component is settable.}\]

\[<\text{void}> \ \text{massRate}(<\text{double}\ \text{value})\ \text{Cumulate to the rate of the mass property of the BasicStandard component.}\]

\[<\text{void}> \ \text{massScale}(<\text{double}\ \text{value})\ \text{Set the scale of the mass property of the BasicStandard component.}\]

\[<\text{void}> \ \text{massSet}(<\text{double}\ \text{value})\ \text{Set the mass property of the basic component.}\]

\[<\text{void}> \ \text{nameFromIndex}(<\text{integer}\ \text{count}>, <\text{varying_string}\ \text{name}>)\ \text{Return the name of a property given is index.}\]

\[<\text{logical}> \ \text{nonEvolvingIsActive}()\ \text{Return whether the nonEvolving implementation of the basic component class is active.}\]

\[<\text{logical}> \ \text{nullIsActive}()\ \text{Return whether the null implementation of the basic component class is active.}\]

\[<\text{void}> \ \text{odeStepRatesInitialize}()\ \text{Initialize rates for evolvable properties.}\]

\[<\text{void}> \ \text{odeStepScalesInitialize}()\ \text{Initialize scales for evolvable properties.}\]

\[<\text{void}> \ \text{output}(<\text{integer}\ \text{integerProperty}>, <\text{integer}\ \text{integerBufferCount}>, <\text{integer}\ [\text{integer}]>, <\text{integer}\ \text{integerBuffer}>, <\text{integer}\ \text{doubleProperty}>, <\text{integer}\ \text{doubleBufferCount}>, <\text{double}\ [\text{double}]), <\text{double}\ \text{doubleBuffer}>, <\text{double}\ \text{time}>, <\text{integer}\ \text{instance}>)\ \text{Generate values of outputtable properties.}\]

\[<\text{void}> \ \text{outputCount}(<\text{integer}\ \text{integerPropertyCount}>, <\text{integer}\ \text{doublePropertyCount}>, <\text{double}\ \text{time}>, <\text{integer}\ \text{instance}>)\ \text{Compute a count of outputtable properties.}\]
15.5. Objects

```c
<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 output-puttable properties.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the basic component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<double> time() Get the time property of the basic component.

<integer> timeCount() Compute the count of evolvable quantities in the time property of the BasicNonEvolving component.

<double> timeIsGettable() Get the time property of the basic component.

<logical> timeIsSettable() Specify whether the time property of the basic component is settable.

<double> timeLastIsolated() Get the timeLastIsolated property of the basic component.

<double> timeLastIsolatedIsGettable() Get the timeLastIsolated property of the basic component.

<logical> timeLastIsolatedIsSettable() Specify whether the timeLastIsolated property of the basic component is settable.

<void> timeLastIsolatedSet(<double> value) Set the timeLastIsolated property of the basic component.

<void> timeRate(<double> value) Cumulate to the rate of the time property of the BasicNonEvolving component.

<void> timeScale(<double> value) Set the scale of the time property of the BasicNonEvolving component.

<void> timeSet(<double> value) Set the time property of the basic component.

<type(varying_string)> type() Return the type of this object.
```
nodeComponentBasicNonEvolving

<double> accretionRate() Get the accretionRate property of the basic component.

<double> accretionRateIsGettable() Get the accretionRate property of the basic component.

<logical> accretionRateIsSettable() Specify whether the accretionRate property of the basic component is settable.

<void> accretionRateSet(<double> value) Set the accretionRate property of the basic component.

<void> builder(<**type(node)>componentDefinition) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius, <componentType> [componentType], <massType> [massType], <weightBy> [weightBy], <integer> [weightIndex], <logical> [haloLoaded]) Compute the mass enclosed within a radius.

**type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<double> mass() Get the mass property of the basic component.

<integer> massCount() Compute the count of evolvable quantities in the mass property of the BasicStandard component.

<double> massIsGettable() Get the mass property of the basic component.

<logical> massIsSettable() Specify whether the mass property of the basic component is settable.
15.5. Objects

<void> massRate(<double> value) Cumulate to the rate of the mass property of the BasicStandard component.

<void> massScale(<double> value) Set the scale of the mass property of the BasicStandard component.

<void> massSet(<double> value) Set the mass property of the basic component.

<void> nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

<logical> nonEvolvingIsActive() Return whether the nonEvolving implementation of the basic component class is active.

<logical> nullIsActive() Return whether the null implementation of the basic component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,::)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,::)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double::> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double::> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double::> array→) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the basic component class is active.
15. Coding Galacticus

```xml
<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<double> time() Get the time property of the basic component.

<integer> timeCount() Compute the count of evolvable quantities in the time property of the BasicNonEvolving component.

<double> timeIsGettable() Get the time property of the basic component.

<logical> timeIsSettable() Specify whether the time property of the basic component is settable.

<double> timeLastIsolated() Get the timeLastIsolated property of the basic component.

<double> timeLastIsolatedIsGettable() Get the timeLastIsolated property of the basic component.

<logical> timeLastIsolatedIsSettable() Specify whether the timeLastIsolated property of the basic component is settable.

<void> timeLastIsolatedSet(<double> value) Set the timeLastIsolated property of the basic component.

<void> timeRate(<double> value) Cumulate to the rate of the time property of the BasicNonEvolving component.

<void> timeScale(<double> value) Set the scale of the time property of the BasicNonEvolving component.

<void> timeSet(<double> value) Set the time property of the basic component.

<type(varying_string)> type() Return the type of this object.

nodeComponentBasicNull

<double> accretionRate() Get the accretionRate property of the basic component.

<double> accretionRateIsGettable() Get the accretionRate property of the basic component.

<logical> accretionRateIsSettable() Specify whether the accretionRate property of the basic component is settable.

<void> accretionRateSet(<double> value) Set the accretionRate property of the basic component.

<void> builder(<**type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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.
```
15.5. Objects

<void> destroy() Destroy the object.
<void> dump() Generate an ASCII dump of all properties.
<void> dumpRaw(<integer> fileHandle) Generate a binary dump of all properties.
<double> enclosedMass(<double> radius →, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.
<typename(treeNode)> host() Return a pointer to the host treeNode object.
<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.
<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.
<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.
<void> initialize() Initialize the object.
<double> mass() Get the mass property of the basic component.
<integer> massCount() Compute the count of evolvable quantities in the mass property of the BasicStandard component.
<double> massIsGettable() Get the mass property of the basic component.
<logical> massIsSettable() Specify whether the mass property of the basic component is settable.
<void> massRate(<double> value) Cumulate to the rate of the mass property of the BasicStandard component.
<void> massScale(<double> value) Set the scale of the mass property of the BasicStandard component.
<void> massSet(<double> value) Set the mass property of the basic component.
<void> nameFromIndex(<integer> count →, <varying_string>name←) Return the name of a property given its index.
<logical> nonEvolvingIsActive() Return whether the nonEvolving implementation of the basic component class is active.
<logical> nullIsActive() Return whether the null implementation of the basic component class is active.
<void> odeStepRatesInitialize() Initialize rates for evolvable properties.
<void> odeStepScalesInitialize() Initialize scales for evolvable properties.
<void> output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.
15. Coding GALACTICUS

<void> outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the basic component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<double> time() Get the time property of the basic component.

<integer> timeCount() Compute the count of evolvable quantities in the time property of the BasicNonEvolving component.

<double> timeIsGettable() Get the time property of the BasicNonEvolving component.

<logical> timeIsSettable() Specify whether the time property of the basic component is settable.

<double> timeLastIsolated() Get the timeLastIsolated property of the basic component.

<double> timeLastIsolatedIsGettable() Get the timeLastIsolated property of the basic component.

<logical> timeLastIsolatedIsSettable() Specify whether the timeLastIsolated property of the basic component is settable.

<void> timeLastIsolatedSet(<double> value) Set the timeLastIsolated property of the basic component.

<void> timeRate(<double> value) Cumulate to the rate of the time property of the BasicNonEvolving component.
15.5. Objects

<void> timeScale(<double> value) Set the scale of the time property of the BasicNonEvolving component.

<void> timeSet(<double> value) Set the time property of the basic component.

<type(varying_string)> type() Return the type of this object.

nodeComponentBasicStandard

<double> accretionRate() Get the accretionRate property of the basic component.

<double> accretionRateIsGettable() Get the accretionRate property of the basic component.

<logical> accretionRateIsSettable() Specify whether the accretionRate property of the basic component is settable.

<void> accretionRateSet(<double> value) Set the accretionRate property of the basic component.

<void> builder(<*type(node)>componentDefinition →) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle →) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the mass enclosed within a radius.

<*type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<double> mass() Get the mass property of the basic component.
\(<\text{integer}>\) massCount() Compute the count of evolvable quantities in the mass property of the BasicStandard component.

\(<\text{double}>\) massIsGettable() Get the mass property of the basic component.

\(<\text{logical}>\) massIsSettable() Specify whether the mass property of the basic component is settable.

\(<\text{void}>\) massRate(<\text{double}> value) Cumulate to the rate of the mass property of the BasicStandard component.

\(<\text{void}>\) massScale(<\text{double}> value) Set the scale of the mass property of the BasicStandard component.

\(<\text{void}>\) massSet(<\text{double}> value) Set the mass property of the basic component.

\(<\text{void}>\) nameFromIndex(<\text{integer}> count →, <\text{varying_string}>name←) Return the name of a property given is index.

\(<\text{logical}>\) nonEvolvingIsActive() Return whether the nonEvolving implementation of the basic component class is active.

\(<\text{logical}>\) nullIsActive() Return whether the null implementation of the basic component class is active.

\(<\text{void}>\) odeStepRatesInitialize() Initialize rates for evolvable properties.

\(<\text{void}>\) odeStepScalesInitialize() Initialize scales for evolvable properties.

\(<\text{void}>\) 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.

\(<\text{void}>\) outputCount(<\text{integer}> integerPropertyCount↔, <\text{integer}> doublePropertyCount↔, <\text{double}> time→, <\text{integer}> instance→) Compute a count of outputtable properties.

\(<\text{void}>\) 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.

\(<\text{double}>\) potential(<\text{double}> radius→, <\text{componentType}> [componentType]→, <\text{massType}> [massType]→, <\text{logical}> [haloLoaded]→) Compute the gravitational potential.

\(<\text{void}>\) readRaw(<\text{integer}> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

\(<\text{double}>\) rotationCurve(<\text{double}> radius→, <\text{componentType}> [componentType]→, <\text{massType}> [massType]→, <\text{logical}> [haloLoaded]→) Compute the rotation curve.

\(<\text{double}>\) rotationCurveGradient(<\text{double}> radius→, <\text{componentType}> [componentType]→, <\text{massType}> [massType]→, <\text{logical}> [haloLoaded]→) Compute the rotation curve gradient.

\(<\text{integer}>\) serializeCount() Return a count of the number of evolvable quantities to be evolved.
15.5. Objects

<void> serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the basic component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the surface density.

<double> time() Get the time property of the basic component.

<integer> timeCount() Compute the count of evolvable quantities in the time property of the BasicNonEvolving component.

<double> timeIsGettable() Get the time property of the basic component.

<logical> timeIsSettable() Specify whether the time property of the basic component is settable.

<double> timeLastIsolated() Get the timeLastIsolated property of the basic component.

<double> timeLastIsolatedIsGettable() Get the timeLastIsolated property of the basic component.

<logical> timeLastIsolatedIsSettable() Specify whether the timeLastIsolated property of the basic component is settable.

<void> timeLastIsolatedSet(<double> value) Set the timeLastIsolated property of the basic component.

<void> timeRate(<double> value) Cumulate to the rate of the time property of the BasicNonEvolving component.

<void> timeScale(<double> value) Set the scale of the time property of the BasicNonEvolving component.

<void> timeSet(<double> value) Set the time property of the basic component.

<type(varying_string)> type() Return the type of this object.

nodeComponentBlackHole

<void> builder(<*type(node)> componentDefinition →) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

 <*>type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<double> mass() Get the mass property of the blackHole component.

<integer> massCount() Compute the count of evolvable quantities in the mass property of the BlackHoleSimple component.

<double> massIsGettable() Get the mass property of the blackHole component.

<logical> massIsSettable() Specify whether the mass property of the blackHole component is settable.

<void> massRate(<double> value) Cumulate to the rate of the mass property of the BlackHoleSimple component.

<void> massScale(<double> value) Set the scale of the mass property of the BlackHoleSimple component.

<double> massSeed() Get the massSeed property of the blackHole component.

<double> massSeedIsGettable() Get the massSeed property of the blackHole component.

<logical> massSeedIsSettable() Specify whether the massSeed property of the blackHole component is settable.

<void> massSet(<double> value) Set the mass property of the blackHole component.

<void> nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the blackHole component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.
15.5. Objects

```null
<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)>
grandBuffer ↔, <integer>doubleProperty ↔, <integer>doubleBufferCount ↔, <double(:,:)>
doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of out-puttable properties.

<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double>
time →, <integer> instance →) Compute a count of outputtable properties.

<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 out-
puttable properties.

<double> potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →,
logical> [haloLoaded] →) Compute the gravitational potential.

<double> radialPosition() Get the radialPosition property of the blackHole component.

<integer> radialPositionCount() Compute the count of evolvable quantities in the radialPosition property of the BlackHoleStandard component.

<double> radialPositionIsGettable() Get the radialPosition property of the blackHole component.

<logical> radialPositionIsSettable() Specify whether the radialPosition property of the blackHole component is settable.

<void> radialPositionRate(<double> value) Cumulate to the rate of the radialPosition property of the BlackHoleStandard component.

<void> radialPositionScale(<double> value) Set the scale of the radialPosition property of the BlackHoleStandard component.

<void> radialPositionSet(<double> value) Set the radialPosition property of the blackHole component.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →,
logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →,
logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

<logical> simpleIsActive() Return whether the simple implementation of the blackHole component class is active.

<double> spin() Get the spin property of the blackHole component.
```
<integer> spinCount() Compute the count of evolvable quantities in the spin property of the BlackHoleStandard component.

<double> spinIsGettable() Get the spin property of the blackHole component.

<logical> spinIsSettable() Specify whether the spin property of the blackHole component is settable.

<void> spinRate(double value) Cumulate to the rate of the spin property of the BlackHoleStandard component.

<void> spinScale(double value) Set the scale of the spin property of the BlackHoleStandard component.

<double> spinSeed() Get the spinSeed property of the blackHole component.

<double> spinSeedIsGettable() Get the spinSeed property of the blackHole component.

<logical> spinSeedIsSettable() Specify whether the spinSeed property of the blackHole component is settable.

<void> spinSet(double value) Set the spin property of the blackHole component.

<logical> standardIsActive() Return whether the standard implementation of the blackHole component class is active.

<double> surfaceDensity(double(3) positionCylindrical→, componentType [componentType]→, massType [massType]→, logical [haloLoaded]→) Compute the surface density.

<double> tripleInteractionTime() Get the tripleInteractionTime property of the blackHole component.

<double> tripleInteractionTimeIsGettable() Get the tripleInteractionTime property of the blackHole component.

<logical> tripleInteractionTimeIsSettable() Specify whether the tripleInteractionTime property of the blackHole component is settable.

<void> tripleInteractionTimeSet(double value) Set the tripleInteractionTime property of the blackHole component.

?type(varying_string)> type() Return the type of this object.

nodeComponentBlackHoleNull

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<double> mass() Get the mass property of the blackHole component.

<integer> massCount() Compute the count of evolvable quantities in the mass property of the BlackHoleSimple component.

<double> massIsGettable() Get the mass property of the blackHole component.

<logical> massIsSettable() Specify whether the mass property of the blackHole component is settable.

<void> massRate(<double> value) Cumulate to the rate of the mass property of the BlackHoleSimple component.

<void> massScale(<double> value) Set the scale of the mass property of the BlackHoleSimple component.

<double> massSeed() Get the massSeed property of the blackHole component.

<double> massSeedIsGettable() Get the massSeed property of the blackHole component.

<logical> massSeedIsSettable() Specify whether the massSeed property of the blackHole component is settable.

<void> massSet(<double> value) Set the mass property of the blackHole component.

<void> nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the blackHole component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.
15. Coding Galacticus

```c
<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)>
    integerBuffer ↔, <integer> doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)>
    doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of oututable properties.

<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double>
    time →, <integer> instance →) Compute a count of oututable properties.

<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 oututable properties.

<double> potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →,
    <logical> [haloLoaded] →) Compute the gravitational potential.

<double> radialPosition() Get the radialPosition property of the blackHole component.

<integer> radialPositionCount() Compute the count of evolvable quantities in the radialPosition property of the BlackHoleStandard component.

<double> radialPositionIsGettable() Get the radialPosition property of the blackHole component.

<logical> radialPositionIsSettable() Specify whether the radialPosition property of the blackHole component is settable.

<void> radialPositionRate(<double> value) Cumulate to the rate of the radialPosition property of the BlackHoleStandard component.

<void> radialPositionScale(<double> value) Set the scale of the radialPosition property of the BlackHoleStandard component.

<void> radialPositionSet(<double> value) Set the radialPosition property of the blackHole component.

<void> readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →,
    <logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →,
    <logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.
```
15.5. Objects

<logical> simpleIsActive() Return whether the simple implementation of the blackHole component class is active.

<double> spin() Get the spin property of the blackHole component.

<integer> spinCount() Compute the count of evolvable quantities in the spin property of the BlackHoleStandard component.

<double> spinIsGettable() Get the spin property of the blackHole component.

<logical> spinIsSettable() Specify whether the spin property of the blackHole component is settable.

<void> spinRate(<double> value) Cumulate to the rate of the spin property of the BlackHoleStandard component.

<void> spinScale(<double> value) Set the scale of the spin property of the BlackHoleStandard component.

<double> spinSeed() Get the spinSeed property of the blackHole component.

<double> spinSeedIsGettable() Get the spinSeed property of the blackHole component.

<logical> spinSeedIsSettable() Specify whether the spinSeed property of the blackHole component is settable.

<void> spinSet(<double> value) Set the spin property of the blackHole component.

<logical> standardIsActive() Return whether the standard implementation of the blackHole component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<double> tripleInteractionTime() Get the tripleInteractionTime property of the blackHole component.

<double> tripleInteractionTimeIsGettable() Get the tripleInteractionTime property of the blackHole component.

<logical> tripleInteractionTimeIsSettable() Specify whether the tripleInteractionTime property of the blackHole component is settable.

<void> tripleInteractionTimeSet(<double> value) Set the tripleInteractionTime property of the blackHole component.

<type(varying_string)> type() Return the type of this object.

nodeComponentBlackHoleSimple

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.
15. Coding GALACTICUS

```c
<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.
<void> dump() Generate an ASCII dump of all properties.
<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.
<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.
<type(treeNode)> host() Return a pointer to the host treeNode object.
<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.
<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.
<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.
<void> initialize() Initialize the object.
<double> mass() Get the mass property of the blackHole component.
<integer> massCount() Compute the count of evolvable quantities in the mass property of the BlackHoleSimple component.
<double> massIsGettable() Get the mass property of the blackHole component.
<logical> massIsSettable() Specify whether the mass property of the blackHole component is settable.
<void> massRate(<double> value) Cumulate to the rate of the mass property of the BlackHoleSimple component.
<void> massScale(<double> value) Set the scale of the mass property of the BlackHoleSimple component.
<double> massSeed() Get the massSeed property of the blackHole component.
<double> massSeedIsGettable() Get the massSeed property of the blackHole component.
<logical> massSeedIsSettable() Specify whether the massSeed property of the blackHole component is settable.
<void> massSet(<double> value) Set the mass property of the blackHole component.
<void> nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.
```
nullIsActive() Return whether the null implementation of the blackHole component class is active.

odeStepRatesInitialize() Initialize rates for evolvable properties.

odeStepScalesInitialize() Initialize scales for evolvable properties.

output(integerProperty ↔, integerBufferCount ↔, integerBuffer ↔, doubleProperty ↔, doubleBufferCount ↔, doubleBuffer ↔, time →, instance →) Generate values of outputtable properties.

outputCount(integerPropertyCount ↔, doublePropertyCount ↔, time →, instance →) Compute a count of outputtable properties.

outputNames(integerProperty ↔, integerPropertyNames ↔, integerPropertyComments ↔, integerPropertyUnitsSI ↔, doubleProperty ↔, doublePropertyNames ↔, doublePropertyComments ↔, doublePropertyUnitsSI ↔, time →, instance →) Generate names of outputtable properties.

potential(radius →, componentType →, massType →, haloLoaded →) Compute the gravitational potential.

radialPosition() Get the radialPosition property of the blackHole component.

radialPositionCount() Compute the count of evolvable quantities in the radialPosition property of the BlackHoleStandard component.

radialPositionIsGettable() Get the radialPosition property of the blackHole component.

radialPositionIsSettable() Specify whether the radialPosition property of the blackHole component is settable.

radialPositionRate(value) Cumulate to the rate of the radialPosition property of the BlackHoleStandard component.

radialPositionScale(value) Set the scale of the radialPosition property of the BlackHoleStandard component.

radialPositionSet(value) Set the radialPosition property of the blackHole component.

readRaw(fileHandle) Read a binary dump of the nodeComponent from the given fileHandle.

rotationCurve(radius →, componentType →, massType →, haloLoaded →) Compute the rotation curve.

rotationCurveGradient(radius →, componentType →, massType →, haloLoaded →) Compute the rotation curve gradient.

serializeCount() Return a count of the number of evolvable quantities to be evolved.

serializeRates(array) Serialize the evolvable rates to an array.
<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<logical> simpleIsActive() Return whether the simple implementation of the blackHole component class is active.

<double> spin() Get the spin property of the blackHole component.

<integer> spinCount() Compute the count of evolvable quantities in the spin property of the BlackHoleStandard component.

<double> spinIsGettable() Get the spin property of the blackHole component.

<logical> spinIsSettable() Specify whether the spin property of the blackHole component is settable.

<void> spinRate(<double> value) Cumulate to the rate of the spin property of the BlackHoleStandard component.

<void> spinScale(<double> value) Set the scale of the spin property of the BlackHoleStandard component.

<double> spinSeed() Get the spinSeed property of the blackHole component.

<double> spinSeedIsGettable() Get the spinSeed property of the blackHole component.

<logical> spinSeedIsSettable() Specify whether the spinSeed property of the blackHole component is settable.

<void> spinSet(<double> value) Set the spin property of the blackHole component.

<logical> standardIsActive() Return whether the standard implementation of the blackHole component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<double> tripleInteractionTime() Get the tripleInteractionTime property of the blackHole component.

<double> tripleInteractionTimeIsGettable() Get the tripleInteractionTime property of the blackHole component.

<logical> tripleInteractionTimeIsSettable() Specify whether the tripleInteractionTime property of the blackHole component is settable.

<void> tripleInteractionTimeSet(<double> value) Set the tripleInteractionTime property of the blackHole component.

<type(varying_string)> type() Return the type of this object.
15.5. Objects

nodeComponentBlackHoleStandard

```cpp
<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<ddouble> 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←) Deserializethe evolvable quantities from an array.

<void> destroy() Destroy the object.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<ddouble> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)
Compute the mass enclosed within a radius.

<*type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulatetotherateofthe hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulatetotherateofthe hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<ddouble> mass() Get the mass property of the blackHole component.

<integer> massCount() Compute the count of evolvable quantities in the mass property of the BlackHoleSimple component.

<ddouble> massIsGettable() Get the mass property of the blackHole component.

<logical> massIsSettable() Specify whether the mass property of the blackHole component is settable.

<void> massRate(<double> value) Cumulate to the rate of the mass property of the BlackHoleSimple component.

<void> massScale(<double> value) Set the scale of the mass property of the BlackHoleSimple component.

<ddouble> massSeed() Get the massSeed property of the blackHole component.

<double> massSeedIsGettable() Get the massSeed property of the blackHole component.
```
15. Coding GALACTICUS

```c
<logical> massSeedIsSettable() Specify whether the massSeed property of the blackHole component is settable.

<void> massSet(<double> value) Set the mass property of the blackHole component.

<void> nameFromIndex(<integer> count, <varying_string>name) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the blackHole component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty, <integer> integerBufferCount, <integer(:,:)_integerBuffer, <integer>doubleProperty, <integer> doubleBufferCount, <double(:,:) doubleBuffer, <double> time, <integer> instance) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount, <integer> doublePropertyCount, <double> time, <integer> instance) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius, <componentType> [componentType], <massType> [massType], <logical> [haloLoaded]) Compute the gravitational potential.

<double> radialPosition() Get the radialPosition property of the blackHole component.

<integer> radialPositionCount() Compute the count of evolvable quantities in the radialPosition property of the BlackHoleStandard component.

<double> radialPositionIsGettable() Get the radialPosition property of the blackHole component.

<logical> radialPositionIsSettable() Specify whether the radialPosition property of the blackHole component is settable.

<void> radialPositionRate(<double> value) Cumulate to the rate of the radialPosition property of the BlackHoleStandard component.

<void> radialPositionScale(<double> value) Set the scale of the radialPosition property of the BlackHoleStandard component.

<void> radialPositionSet(<double> value) Set the radialPosition property of the blackHole component.

<void> readRaw(<integer> fileHandle) Read a binary dump of the nodeComponent from the given fileHandle.
```
15.5. Objects

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double():> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double():> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double():> array →) Serialize the evolvable quantities to an array.

logical> simpleIsActive() Return whether the simple implementation of the blackHole component class is active.

<double> spin() Get the spin property of the blackHole component.

<integer> spinCount() Compute the count of evolvable quantities in the spin property of the BlackHoleStandard component.

<double> spinIsGettable() Get the spin property of the blackHole component.

<logical> spinIsSettable() Specify whether the spin property of the blackHole component is settable.

<void> spinRate(<double> value) Cumulate to the rate of the spin property of the BlackHoleStandard component.

<void> spinScale(<double> value) Set the scale of the spin property of the BlackHoleStandard component.

<double> spinSeed() Get the spinSeed property of the blackHole component.

<double> spinSeedIsGettable() Get the spinSeed property of the blackHole component.

<logical> spinSeedIsSettable() Specify whether the spinSeed property of the blackHole component is settable.

<void> spinSet(<double> value) Set the spin property of the blackHole component.

<logical> standardIsActive() Return whether the standard implementation of the blackHole component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the surface density.

<double> tripleInteractionTime() Get the tripleInteractionTime property of the blackHole component.

<double> tripleInteractionTimeIsGettable() Get the tripleInteractionTime property of the blackHole component.

<logical> tripleInteractionTimeIsSettable() Specify whether the tripleInteractionTime property of the blackHole component is settable.

<void> tripleInteractionTimeSet(<double> value) Set the tripleInteractionTime property of the blackHole component.

<type(varying_string)> type() Return the type of this object.
nodeComponentDarkMatterProfile
<void> builder(<type(node)> componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count→, <varying_string> name←) Return the name of a property given its index.

logical nullIsActive() Return whether the null implementation of the darkMatterProfile component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty←, <integer> integerBufferCount←, <integer(:,:)> integerBuffer←, <integer> doubleProperty←, <integer> doubleBufferCount←, <double(:,:)> doubleBuffer←, <double> time→, <integer> instance→) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount←, <integer> doublePropertyCount←, <double> time→, <integer> instance→) Compute a count of outputtable properties.
15.5. Objects

<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.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<double> scale() Get the scale property of the darkMatterProfile component.

<integer> scaleCount() Compute the count of evolvable quantities in the scale property of the DarkMatterProfileScale component.

<void> scaleFunction(<function()> deferredFunction) Set the function to be used for the get method of the scale property of the darkMatterProfile component.

<double> scaleGrowthRate() Get the scaleGrowthRate property of the darkMatterProfile component.

<double> scaleGrowthRateIsGettable() Get the scaleGrowthRate property of the darkMatterProfile component.

<logical> scaleGrowthRateIsSettable() Specify whether the scaleGrowthRate property of the darkMatterProfile component is settable.

<void> scaleGrowthRateSet(<double> value) Set the scaleGrowthRate property of the darkMatterProfile component.

<logical> scaleIsActive() Return whether the scale implementation of the darkMatterProfile component class is active.

<logical> scaleIsAttached() Return whether the get method of the scale property of the darkMatterProfile component has been attached to a function.

<double> scaleIsGettable() Get the scale property of the darkMatterProfile component.

<logical> scaleIsSettable() Specify whether the scale property of the darkMatterProfile component is settable.

<void> scaleRate(<double> value) Cumulate to the rate of the scale property of the DarkMatterProfileScale component.

<void> scaleScale(<double> value) Set the scale of the scale property of the DarkMatterProfileScale component.

<void> scaleSet(<double> value) Set the scale property of the darkMatterProfile component.

<logical> scaleShapeIsActive() Return whether the scaleShape implementation of the darkMatterProfile component class is active.
15. Coding GALACTICUS

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<double> shape() Get the shape property of the darkMatterProfile component.

<integer> shapeCount() Compute the count of evolvable quantities in the shape property of the DarkMatterProfileScaleShape component.

<double> shapeGrowthRate() Get the shapeGrowthRate property of the darkMatterProfile component.

<double> shapeGrowthRateIsGettable() Get the shapeGrowthRate property of the darkMatterProfile component.

<logical> shapeGrowthRateIsSettable() Specify whether the shapeGrowthRate property of the darkMatterProfile component is settable.

<void> shapeGrowthRateSet(<double> value) Set the shapeGrowthRate property of the darkMatterProfile component.

<double> shapeIsGettable() Get the shape property of the darkMatterProfile component.

<logical> shapeIsSettable() Specify whether the shape property of the darkMatterProfile component is settable.

<void> shapeRate(<double> value) Cumulate to the rate of the shape property of the DarkMatterProfileScaleShape component.

<void> shapeScale(<double> value) Set the scale of the shape property of the DarkMatterProfileScaleShape component.

<void> shapeSet(<double> value) Set the shape property of the darkMatterProfile component.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

nodeComponentDarkMatterProfileNull

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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.
15.5. Objects

<void> destroy() Destroy the object.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<type(treeNode)> host() Return a pointer to the host TreeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

logical> nullIsActive() Return whether the null implementation of the darkMatterProfile component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer>(::) integerBuffer↔, <integer> doubleProperty↔, <integer> doubleBufferCount↔, <double>(::) doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.
<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

<double> scale() Get the scale property of the darkMatterProfile component.

<integer> scaleCount() Compute the count of evolvable quantities in the scale property of the DarkMatterProfileScale component.

<void> scaleFunction(<function()> deferredFunction) Set the function to be used for the get method of the scale property of the darkMatterProfile component.

<double> scaleGrowthRate() Get the scaleGrowthRate property of the darkMatterProfile component.

<double> scaleGrowthRateIsGettable() Get the scaleGrowthRate property of the darkMatterProfile component.

<logical> scaleGrowthRateIsSettable() Specify whether the scaleGrowthRate property of the darkMatterProfile component is settable.

<void> scaleGrowthRateSet(<double> value) Set the scaleGrowthRate property of the darkMatterProfile component.

<logical> scaleIsActive() Return whether the scale implementation of the darkMatterProfile component class is active.

<logical> scaleIsAttached() Return whether the get method of the scale property of the darkMatterProfile component has been attached to a function.

<double> scaleIsGettable() Get the scale property of the darkMatterProfile component.

<logical> scaleIsSettable() Specify whether the scale property of the darkMatterProfile component is settable.

<void> scaleRate(<double> value) Cumulate to the rate of the scale property of the DarkMatterProfileScale component.

<void> scaleScale(<double> value) Set the scale of the scale property of the DarkMatterProfileScale component.

<void> scaleSet(<double> value) Set the scale property of the darkMatterProfile component.

<logical> scaleShapeIsActive() Return whether the scaleShape implementation of the darkMatterProfile component class is active.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

<double> shape() Get the shape property of the darkMatterProfile component.

<integer> shapeCount() Compute the count of evolvable quantities in the shape property of the DarkMatterProfileScaleShape component.
15.5. Objects

<double> shapeGrowthRate() Get the shapeGrowthRate property of the darkMatterProfile component.

<double> shapeGrowthIsGettable() Get the shapeGrowthRate property of the darkMatterProfile component.

<logical> shapeGrowthIsSettable() Specify whether the shapeGrowthRate property of the darkMatterProfile component is settable.

<void> shapeGrowthSet(<double> value) Set the shapeGrowthRate property of the darkMatterProfile component.

<double> shapeIsGettable() Get the shape property of the darkMatterProfile component.

<logical> shapeIsSettable() Specify whether the shape property of the darkMatterProfile component is settable.

<void> shapeRate(<double> value) Cumulate to the rate of the shape property of the DarkMatterProfileScaleShape component.

<void> shapeScale(<double> value) Set the scale of the shape property of the DarkMatterProfileScaleShape component.

<void> shapeSet(<double> value) Set the shape property of the darkMatterProfile component.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

type(varying_string) type() Return the type of this object.

nodeComponentDarkMatterProfileScale

<void> builder(<*type(node)> componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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←) Deserializethe evolvable quantities from an array.

<void> destroy() Destroy the object.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<*type(treeNode)> host() Return a pointer to the host treeNode object.
15. Coding GALACTICUS

```cpp
// hotHaloCoolingAbundancesRate(type(abundances) value) Cumulate to the rate of the
// hotHaloCoolingAbundances property of the hotHalo component.

// hotHaloCoolingAngularMomentumRate(double value) Cumulate to the rate of the hotHaloCoolingAngularMomentum
// property of the hotHalo component.

// hotHaloCoolingMassRate(double value) Cumulate to the rate of the hotHaloCoolingMass
// property of the hotHalo component.

// initialize() Initialize the object.

// nameFromIndex(integer count, varying_string name) Return the name of a property given is index.

// nullIsActive() Return whether the null implementation of the darkMatterProfile component class is active.

// odeStepRatesInitialize() Initialize rates for evolvable properties.

// odeStepScalesInitialize() Initialize scales for evolvable properties.

// output(integer integerProperty, integerBufferCount, integerBuffer, doubleProperty, doubleBufferCount, doubleBuffer,
// double time, integer instance) Generate values of outputtable properties.

// outputCount(integer integerPropertyCount, doublePropertyCount, double time, integer instance) Compute a count of outputtable properties.

// outputNames(integer integerProperty, integerPropertyNames, integerPropertyComments, integerPropertyUnitsSI, doubleProperty, doublePropertyNames, doublePropertyComments, doublePropertyUnitsSI, double time, integer instance) Generate names of outputtable properties.

// potential(double radius, componentType [componentType] [massType] [haloLoaded]) Compute the gravitational potential.

// readRaw(integer fileHandle) Read a binary dump of the nodeComponent from the given fileHandle.

// rotationCurve(double radius, componentType [componentType] [massType] [haloLoaded]) Compute the rotation curve.

// rotationCurveGradient(double radius, componentType [componentType] [massType] [haloLoaded]) Compute the rotation curve gradient.

// scale() Get the scale property of the darkMatterProfile component.

// scaleCount() Compute the count of evolvable quantities in the scale property of the DarkMatterProfileScale component.

// scaleFunction(function() deferredFunction) Set the function to be used for the get method of the scale property of the darkMatterProfile component.
```
<double> scaleGrowthRate() Get the scaleGrowthRate property of the darkMatterProfile component.

<double> scaleGrowthRateIsGettable() Get the scaleGrowthRate property of the darkMatterProfile component.

<logical> scaleGrowthRateIsSettable() Specify whether the scaleGrowthRate property of the darkMatterProfile component is settable.

<void> scaleGrowthRateSet(<double> value) Set the scaleGrowthRate property of the darkMatterProfile component.

<logical> scaleIsActive() Return whether the scale implementation of the darkMatterProfile component class is active.

<logical> scaleIsAttached() Return whether the get method of the scale property of the darkMatterProfile component has been attached to a function.

<double> scaleIsGettable() Get the scale property of the darkMatterProfile component.

<logical> scaleIsSettable() Specify whether the scale property of the darkMatterProfile component is settable.

<void> scaleRate(<double> value) Cumulate to the rate of the scale property of the DarkMatterProfileScale component.

<void> scaleScale(<double> value) Set the scale of the scale property of the DarkMatterProfileScale component.

<void> scaleSet(<double> value) Set the scale property of the darkMatterProfile component.

<logical> scaleShapeIsActive() Return whether the scaleShape implementation of the darkMatterProfile component class is active.

<double> scaleValue() Get the scale property of the darkMatterProfile component.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<double> shape() Get the shape property of the darkMatterProfile component.

<integer> shapeCount() Compute the count of evolvable quantities in the shape property of the DarkMatterProfileScaleShape component.

<double> shapeGrowthRate() Get the shapeGrowthRate property of the darkMatterProfile component.

<double> shapeGrowthRateIsGettable() Get the shapeGrowthRate property of the darkMatterProfile component.

<logical> shapeGrowthRateIsSettable() Specify whether the shapeGrowthRate property of the darkMatterProfile component is settable.
15. Coding Galacticus

```c
<void> shapeGrowthRateSet(<double> value) Set the shapeGrowthRate property of the darkMatterProfile component.

<double> shapeIsGettable() Get the shape property of the darkMatterProfile component.

<logical> shapeIsSettable() Specify whether the shape property of the darkMatterProfile component is settable.

<void> shapeRate(<double> value) Cumulate to the rate of the shape property of the DarkMatterProfileScaleShape component.

<void> shapeScale(<double> value) Set the scale of the shape property of the DarkMatterProfileScaleShape component.

<void> shapeSet(<double> value) Set the shape property of the darkMatterProfile component.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.
```

```c
nodeComponentDarkMatterProfileScaleShape

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<*type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.
```
15.5. Objects

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count →, <varying_string> name ←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the darkMatterProfile component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:) > integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:) > doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

<void> readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

<double> scale() Get the scale property of the darkMatterProfile component.

<integer> scaleCount() Compute the count of evolvable quantities in the scale property of the DarkMatterProfileScale component.

<void> scaleFunction(<function()> deferredFunction) Set the function to be used for the get method of the scale property of the darkMatterProfile component.

<double> scaleGrowthRate() Get the scaleGrowthRate property of the darkMatterProfile component.

<double> scaleGrowthRateIsGettable() Get the scaleGrowthRate property of the darkMatterProfile component.

<logical> scaleGrowthRateIsSettable() Specify whether the scaleGrowthRate property of the darkMatterProfile component is settable.
<void> scaleGrowthRateSet(<double> value) Set the scaleGrowthRate property of the darkMatterProfile component.

<logical> scaleIsActive() Return whether the scale implementation of the darkMatterProfile component class is active.

<logical> scaleIsAttached() Return whether the get method of the scale property of the darkMatterProfile component has been attached to a function.

<double> scaleIsGettable() Get the scale property of the darkMatterProfile component.

<logical> scaleIsSettable() Specify whether the scale property of the darkMatterProfile component is settable.

<void> scaleRate(<double> value) Cumulate to the rate of the scale property of the DarkMatterProfileScale component.

<void> scaleScale(<double> value) Set the scale of the scale property of the DarkMatterProfileScale component.

<void> scaleSet(<double> value) Set the scale property of the darkMatterProfile component.

<logical> scaleShapeIsActive() Return whether the scaleShape implementation of the darkMatterProfile component class is active.

<double> scaleValue() Get the scale property of the darkMatterProfile component.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<double> shape() Get the shape property of the darkMatterProfile component.

<integer> shapeCount() Compute the count of evolvable quantities in the shape property of the DarkMatterProfileScaleShape component.

<double> shapeGrowthRate() Get the shapeGrowthRate property of the darkMatterProfile component.

<double> shapeGrowthRateIsGettable() Get the shapeGrowthRate property of the darkMatterProfile component.

<logical> shapeGrowthRateIsSettable() Specify whether the shapeGrowthRate property of the darkMatterProfile component is settable.

<void> shapeGrowthRateSet(<double> value) Set the shapeGrowthRate property of the darkMatterProfile component.

<double> shapeIsGettable() Get the shape property of the darkMatterProfile component.

<logical> shapeIsSettable() Specify whether the shape property of the darkMatterProfile component is settable.
15.5. Objects

<void> shapeRate(<double> value) Cumulate to the rate of the shape property of the DarkMatterProfileScaleShape component.

<void> shapeScale(<double> value) Set the scale of the shape property of the DarkMatterProfileScaleShape component.

<void> shapeSet(<double> value) Set the shape property of the darkMatterProfile component.

<double> surfaceDensity(<double(3)> positionCylindrical, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

nodeComponentDisk

<type(abundances)> abundancesGas() Get the abundancesGas property of the disk component.

<integer> abundancesGasCount() Compute the count of evolvable quantities in the abundancesGas property of the DiskExponential component.

<type(abundances)> abundancesGasIsGettable() Get the abundancesGas property of the disk component.

<logical> abundancesGasIsSettable() Specify whether the abundancesGas property of the disk component is settable.

<void> abundancesGasRate(<type(abundances)> value) Cumulate to the rate of the abundancesGas property of the DiskExponential component.

<void> abundancesGasScale(<type(abundances)> value) Set the scale of the abundancesGas property of the DiskExponential component.

<void> abundancesGasSet(<type(abundances)> value) Set the abundancesGas property of the disk component.

<type(abundances)> abundancesStellar() Get the abundancesStellar property of the disk component.

<integer> abundancesStellarCount() Compute the count of evolvable quantities in the abundancesStellar property of the DiskExponential component.

<type(abundances)> abundancesStellarIsGettable() Get the abundancesStellar property of the disk component.

<logical> abundancesStellarIsSettable() Specify whether the abundancesStellar property of the disk component is settable.

<void> abundancesStellarRate(<type(abundances)> value) Cumulate to the rate of the abundancesStellar property of the DiskExponential component.

<void> abundancesStellarScale(<type(abundances)> value) Set the scale of the abundancesStellar property of the DiskExponential component.

<void> abundancesStellarSet(<type(abundances)> value) Set the abundancesStellar property of the disk component.
15. Coding GALACTICUS

```c
< double > angularMomentum() Get the angularMomentum property of the disk component.

< integer > angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the DiskExponential component.

< double > angularMomentumIsGettable() Get the angularMomentum property of the disk component.

< logical > angularMomentumIsSettable() Specify whether the angularMomentum property of the disk component is settable.

< void > angularMomentumRate(< double > value) Cumulate to the rate of the angularMomentum property of the DiskExponential component.

< void > angularMomentumScale(< double > value) Set the scale of the angularMomentum property of the DiskExponential component.

< void > angularMomentumSet(< double > value) Set the angularMomentum property of the disk component.

< void > builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

< double > 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.

< void > dump() Generate an ASCII dump of all properties.

< void > dumpRaw(< integer > fileHandle→) Generate a binary dump of all properties.

< double > enclosedMass(< double > radius→, < componentType > [componentType]→, < massType > [massType]→, < weightBy > [weightBy]→, < integer > [weightIndex]→, < logical > [haloLoaded]→) Compute the mass enclosed within a radius.

< logical > exponentialIsActive() Return whether the exponential implementation of the disk component class is active.

< double > halfMassRadius() Get the halfMassRadius property of the disk component.

< double > halfMassRadiusIsGettable() Get the halfMassRadius property of the disk component.

< logical > halfMassRadiusIsSettable() Specify whether the halfMassRadius property of the disk component is settable.

<*type(treeNode)> host() Return a pointer to the host treeNode object.

< void > hotHaloCoolingAbundancesRate(< type(abundances) > value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.
```
15.5. Objects

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<logical> isInitialized() Get the isInitialized property of the disk component.

<logical> isInitializedIsGettable() Get the isInitialized property of the disk component.

<logical> isInitializedIsSettable() Specify whether the isInitialized property of the disk component is settable.

<void> isInitializedSet(<logical> value) Set the isInitialized property of the disk component.

<double(:)> luminositiesStellar() Get the luminositiesStellar property of the disk component.

<integer> luminositiesStellarCount() Compute the count of evolvable quantities in the luminositiesStellar property of the DiskExponential component.

<double(:)> luminositiesStellarIsGettable() Get the luminositiesStellar property of the disk component.

<logical> luminositiesStellarIsSettable() Specify whether the luminositiesStellar property of the disk component is settable.

<void> luminositiesStellarRate(<double(:)> value) Cumulate to the rate of the luminositiesStellar property of the DiskExponential component.

<void> luminositiesStellarScale(<double(:)> value) Set the scale of the luminositiesStellar property of the DiskExponential component.

<void> luminositiesStellarSet(<double(:)> value) Set the luminositiesStellar property of the disk component.

<double> massGas() Get the massGas property of the disk component.

<integer> massGasCount() Compute the count of evolvable quantities in the massGas property of the DiskVerySimple component.

<double> massGasIsGettable() Get the massGas property of the disk component.

<logical> massGasIsSettable() Specify whether the massGas property of the disk component is settable.

<void> massGasRate(<double> value) Cumulate to the rate of the massGas property of the DiskVerySimple component.

<void> massGasScale(<double> value) Set the scale of the massGas property of the DiskVerySimple component.

<void> massGasSet(<double> value) Set the massGas property of the disk component.
15. Coding GALACTICUS

<double> massStellar() Get the massStellar property of the disk component.

<integer> massStellarCount() Compute the count of evolvable quantities in the massStellar property of the DiskVerySimple component.

<double> massStellarIsGettable() Get the massStellar property of the disk component.

<logical> massStellarIsSettable() Specify whether the massStellar property of the disk component is settable.

<void> massStellarRate(<double> value) Cumulate to the rate of the massStellar property of the DiskVerySimple component.

<void> massStellarScale(<double> value) Set the scale of the massStellar property of the DiskVerySimple component.

<void> massStellarSet(<double> value) Set the massStellar property of the disk component.

<void> nameFromIndex(<integer> count →, <varying_string>name ←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the disk component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

<double> radius() Get the radius property of the disk component.

<double> radiusIsGettable() Get the radius property of the disk component.

<logical> radiusIsSettable() Specify whether the radius property of the disk component is settable.

<void> radiusSet(<double> value) Set the radius property of the disk component.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.
15.5. Objects

```csharp
<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

<type(history)> starFormationHistory() Get the starFormationHistory property of the disk component.

<integer> starFormationHistoryCount() Compute the count of evolvable quantities in the starFormationHistory property of the DiskExponential component.

<type(history)> starFormationHistoryIsGettable() Get the starFormationHistory property of the disk component.

<logical> starFormationHistoryIsSettable() Specify whether the starFormationHistory property of the disk component is settable.

<void> starFormationHistoryRate(<type(history)> value) Cumulate to the rate of the starFormationHistory property of the DiskExponential component.

<void> starFormationHistoryScale(<type(history)> value) Set the scale of the starFormationHistory property of the DiskExponential component.

<void> starFormationHistorySet(<type(history)> value) Set the starFormationHistory property of the disk component.

<double> starFormationRate() Get the starFormationRate property of the disk component.

<void> starFormationRateFunction(<function()> deferredFunction) Set the function to be used for the get method of the starFormationRate property of the disk component.

<logical> starFormationRateIsAttached() Return whether the get method of the starFormationRate property of the disk component has been attached to a function.

<double> starFormationRateIsGettable() Get the starFormationRate property of the disk component.

<logical> starFormationRateIsSettable() Specify whether the starFormationRate property of the disk component is settable.

<type(history)> stellarPropertiesHistory() Get the stellarPropertiesHistory property of the disk component.

<integer> stellarPropertiesHistoryCount() Compute the count of evolvable quantities in the stellarPropertiesHistory property of the DiskExponential component.

<type(history)> stellarPropertiesHistoryIsGettable() Get the stellarPropertiesHistory property of the disk component.

<logical> stellarPropertiesHistoryIsSettable() Specify whether the stellarPropertiesHistory property of the disk component is settable.
```
\texttt{<void> stellarPropertiesHistoryRate(<type(history)> value)} Cumulate to the rate of the \texttt{stellarPropertiesHistory} property of the \texttt{DiskExponential} component.

\texttt{<void> stellarPropertiesHistoryScale(<type(history)> value)} Set the scale of the \texttt{stellarPropertiesHistory} property of the \texttt{DiskExponential} component.

\texttt{<void> stellarPropertiesHistorySet(<type(history)> value)} Set the \texttt{stellarPropertiesHistory} property of the disk component.

\texttt{<double> surfaceDensity(<double(3)> positionCylindrical \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow)} Compute the surface density.

\texttt{<type(varying_string)> type()} Return the type of this object.

\texttt{<double> velocity()} Get the velocity property of the disk component.

\texttt{<double> velocityIsGettable()} Get the velocity property of the disk component.

\texttt{<logical> velocityIsSettable()} Specify whether the velocity property of the disk component is settable.

\texttt{<void> velocitySet(<double> value)} Set the velocity property of the disk component.

\texttt{<logical> verySimpleIsActive()} Return whether the verySimple implementation of the disk component class is active.

\texttt{nodeComponentDiskExponential}

\texttt{<type(abundances)> abundancesGas()} Get the \texttt{abundancesGas} property of the disk component.

\texttt{<integer> abundancesGasCount()} Compute the count of evolvable quantities in the \texttt{abundancesGas} property of the \texttt{DiskExponential} component.

\texttt{<type(abundances)> abundancesGasIsGettable()} Get the \texttt{abundancesGas} property of the disk component.

\texttt{<logical> abundancesGasIsSettable()} Specify whether the \texttt{abundancesGas} property of the disk component is settable.

\texttt{<void> abundancesGasRate(<type(abundances)> value)} Cumulate to the rate of the \texttt{abundancesGas} property of the \texttt{DiskExponential} component.

\texttt{<void> abundancesGasScale(<type(abundances)> value)} Set the scale of the \texttt{abundancesGas} property of the \texttt{DiskExponential} component.

\texttt{<void> abundancesGasSet(<type(abundances)> value)} Set the \texttt{abundancesGas} property of the disk component.

\texttt{<type(abundances)> abundancesStellar()} Get the \texttt{abundancesStellar} property of the disk component.

\texttt{<integer> abundancesStellarCount()} Compute the count of evolvable quantities in the \texttt{abundancesStellar} property of the \texttt{DiskExponential} component.

\texttt{<type(abundances)> abundancesStellarIsGettable()} Get the \texttt{abundancesStellar} property of the disk component.
15.5. Objects

<logical> abundancesStellarIsSettable() Specify whether the abundancesStellar property of the disk component is settable.

<void> abundancesStellarRate(<type(abundances)> value) Cumulate to the rate of the abundancesStellar property of the DiskExponential component.

<void> abundancesStellarScale(<type(abundances)> value) Set the scale of the abundancesStellar property of the DiskExponential component.

<void> abundancesStellarSet(<type(abundances)> value) Set the abundancesStellar property of the disk component.

<double> angularMomentum() Get the angularMomentum property of the disk component.

<integer> angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the DiskExponential component.

<double> angularMomentumIsGettable() Get the angularMomentum property of the disk component.

<logical> angularMomentumIsSettable() Specify whether the angularMomentum property of the disk component is settable.

<void> angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the DiskExponential component.

<void> angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the DiskExponential component.

<void> angularMomentumSet(<double> value) Set the angularMomentum property of the disk component.

<void> attachPipes() Attach pipes to the exponential disk component.

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.
15. Coding GALACTICUS

<logical> exponentialIsActive() Return whether the exponential implementation of the disk component class is active.

<double> halfMassRadius() Get the halfMassRadius property of the disk component.

<double> halfMassRadiusIsGettable() Get the halfMassRadius property of the disk component.

<logical> halfMassRadiusIsSettable() Specify whether the halfMassRadius property of the disk component is settable.

<type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<logical> isInitialized() Get the isInitialized property of the disk component.

<logical> isInitializedIsGettable() Get the isInitialized property of the disk component.

<logical> isInitializedIsSettable() Specify whether the isInitialized property of the disk component is settable.

<void> isInitializedSet(<logical> value) Set the isInitialized property of the disk component.

<double(:)> luminositiesStellar() Get the luminositiesStellar property of the disk component.

<integer> luminositiesStellarCount() Compute the count of evolvable quantities in the luminositiesStellar property of the DiskExponential component.

<double(:)> luminositiesStellarIsGettable() Get the luminositiesStellar property of the disk component.

<logical> luminositiesStellarIsSettable() Specify whether the luminositiesStellar property of the disk component is settable.

<void> luminositiesStellarRate(<double(:)> value) Cumulate to the rate of the luminositiesStellar property of the DiskExponential component.

<void> luminositiesStellarScale(<double(:)> value) Set the scale of the luminositiesStellar property of the DiskExponential component.

<void> luminositiesStellarSet(<double(:)> value) Set the luminositiesStellar property of the disk component.

<double> massGas() Get the massGas property of the disk component.
15.5. Objects

**<integer> massGasCount()** Compute the count of evolvable quantities in the massGas property of the DiskVerySimple component.

**<double> massGasIsGettable()** Get the massGas property of the disk component.

**<logical> massGasIsSettable()** Specify whether the massGas property of the disk component is settable.

**<void> massGasRate(<double> value)** Cumulate to the rate of the massGas property of the DiskVerySimple component.

**<void> massGasScale(<double> value)** Set the scale of the massGas property of the DiskVerySimple component.

**<void> massGasSet(<double> value)** Set the massGas property of the disk component.

**<double> massStellar()** Get the massStellar property of the disk component.

**<integer> massStellarCount()** Compute the count of evolvable quantities in the massStellar property of the DiskVerySimple component.

**<double> massStellarIsGettable()** Get the massStellar property of the disk component.

**<logical> massStellarIsSettable()** Specify whether the massStellar property of the disk component is settable.

**<void> massStellarRate(<double> value)** Cumulate to the rate of the massStellar property of the DiskVerySimple component.

**<void> massStellarScale(<double> value)** Set the scale of the massStellar property of the DiskVerySimple component.

**<void> massStellarSet(<double> value)** Set the massStellar property of the disk component.

**<void> nameFromIndex(<integer> count→, <varying_string>name←)** Return the name of a property given is index.

**<logical> nullIsActive()** Return whether the null implementation of the disk component class is active.

**<void> odeStepRatesInitialize()** Initialize rates for evolvable properties.

**<void> odeStepScalesInitialize()** Initialize scales for evolvable properties.

**<void> output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→)** Generate values of outputtable properties.

**<void> outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→)** Compute a count of outputtable properties.

**<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.
15. Coding GALACTICUS

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<double> radius() Get the radius property of the disk component.

<double> radiusIsGettable() Get the radius property of the disk component.

<logical> radiusIsSettable() Specify whether the radius property of the disk component is settable.

<void> radiusSet(<double> value) Set the radius property of the disk component.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<type(history)> starFormationHistory() Get the starFormationHistory property of the disk component.

<integer> starFormationHistoryCount() Compute the count of evolvable quantities in the starFormationHistory property of the DiskExponential component.

<type(history)> starFormationHistoryIsGettable() Get the starFormationHistory property of the disk component.

<logical> starFormationHistoryIsSettable() Specify whether the starFormationHistory property of the disk component is settable.

<void> starFormationHistoryRate(<type(history)> value) Cumulate to the rate of the starFormationHistory property of the DiskExponential component.

<void> starFormationHistoryScale(<type(history)> value) Set the scale of the starFormationHistory property of the DiskExponential component.

<void> starFormationHistorySet(<type(history)> value) Set the starFormationHistory property of the disk component.

<double> starFormationRate() Get the starFormationRate property of the disk component.

<void> starFormationRateFunction(<function()> deferredFunction) Set the function to be used for the get method of the starFormationRate property of the disk component.

<logical> starFormationRateIsAttached() Return whether the get method of the starFormationRate property of the disk component has been attached to a function.
15.5. Objects

<double> starFormationRateIsGettable() Get the starFormationRate property of the disk component.

<logical> starFormationRateIsSettable() Specify whether the starFormationRate property of the disk component is settable.

?type(history)> stellarPropertiesHistory() Get the stellarPropertiesHistory property of the disk component.

<integer> stellarPropertiesHistoryCount() Compute the count of evolvable quantities in the stellarPropertiesHistory property of the DiskExponential component.

?type(history)> stellarPropertiesHistoryIsGettable() Get the stellarPropertiesHistory property of the disk component.

<logical> stellarPropertiesHistoryIsSettable() Specify whether the stellarPropertiesHistory property of the disk component is settable.

<void> stellarPropertiesHistoryRate(<type(history)> value) Cumulate to the rate of the stellarPropertiesHistory property of the DiskExponential component.

<void> stellarPropertiesHistoryScale(<type(history)> value) Set the scale of the stellarPropertiesHistory property of the DiskExponential component.

<void> stellarPropertiesHistorySet(<type(history)> value) Set the stellarPropertiesHistory property of the disk component.

<double> surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the surface density.

?type(varying_string)> type() Return the type of this object.

<double> velocity() Get the velocity property of the disk component.

<double> velocityIsGettable() Get the velocity property of the disk component.

<logical> velocityIsSettable() Specify whether the velocity property of the disk component is settable.

<void> velocitySet(<double> value) Set the velocity property of the disk component.

<logical> verySimpleIsActive() Return whether the verySimple implementation of the disk component class is active.

nodeComponentDiskNull

?type(abundances)> abundancesGas() Get the abundancesGas property of the disk component.

<integer> abundancesGasCount() Compute the count of evolvable quantities in the abundancesGas property of the DiskExponential component.

?type(abundances)> abundancesGasIsGettable() Get the abundancesGas property of the disk component.

<logical> abundancesGasIsSettable() Specify whether the abundancesGas property of the disk component is settable.
15. Coding GALACTICUS

```csharp
<void> abundancesGasRate(<type(abundances)> value) Cumulate to the rate of the abundancesGas

property of the DiskExponential component.

<void> abundancesGasScale(<type(abundances)> value) Set the scale of the abundancesGas prop-

erty of the DiskExponential component.

<void> abundancesGasSet(<type(abundances)> value) Set the abundancesGas property of the disk

component.

<type(abundances)> abundancesStellar() Get the abundancesStellar property of the disk com-

ponent.

<integer> abundancesStellarCount() Compute the count of evolvable quantities in the abundancesStellar

property of the DiskExponential component.

<type(abundances)> abundancesStellarIsGettable() Get the abundancesStellar property of the disk

component.

<logical> abundancesStellarIsSettable() Specify whether the abundancesStellar property of the disk

component is settable.

<void> abundancesStellarRate(<type(abundances)> value) Cumulate to the rate of the abundancesStellar

property of the DiskExponential component.

<void> abundancesStellarScale(<type(abundances)> value) Set the scale of the abundancesStellar prop-

erty of the DiskExponential component.

<void> abundancesStellarSet(<type(abundances)> value) Set the abundancesStellar property of the disk

component.

<double> angularMomentum() Get the angularMomentum property of the disk component.

<integer> angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum

property of the DiskExponential component.

<double> angularMomentumIsGettable() Get the angularMomentum property of the disk component.

<logical> angularMomentumIsSettable() Specify whether the angularMomentum property of the disk

component is settable.

<void> angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum prop-

erty of the DiskExponential component.

<void> angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the

DiskExponential component.

<void> angularMomentumSet(<double> value) Set the angularMomentum property of the disk com-

ponent.

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied

XML definition.

<double> density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType>

```
15.5. Objects

<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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle ←) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →)
Compute the mass enclosed within a radius.

<logical> exponentialIsActive() Return whether the exponential implementation of the disk component class is active.

<double> halfMassRadius() Get the halfMassRadius property of the disk component.

<double> halfMassRadiusIsGettable() Get the halfMassRadius property of the disk component.

<logical> halfMassRadiusIsSettable() Specify whether the halfMassRadius property of the disk component is settable.

<*type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<logical> isInitialized() Get the isInitialized property of the disk component.

<logical> isInitializedIsGettable() Get the isInitialized property of the disk component.

<logical> isInitializedIsSettable() Specify whether the isInitialized property of the disk component is settable.

<void> isInitializedSet(<logical> value) Set the isInitialized property of the disk component.

<double(:)> luminositiesStellar() Get the luminositiesStellar property of the disk component.

<integer> luminositiesStellarCount() Compute the count of evolvable quantities in the luminositiesStellar property of the DiskExponential component.

<double(:)> luminositiesStellarIsGettable() Get the luminositiesStellar property of the disk component.
<logical> luminositiesStellarIsSettable() Specify whether the luminositiesStellar property of the disk component is settable.

<void> luminositiesStellarRate(<double(:)> value) Cumulate to the rate of the luminositiesStellar property of the DiskExponential component.

<void> luminositiesStellarScale(<double(:)> value) Set the scale of the luminositiesStellar property of the DiskExponential component.

<void> luminositiesStellarSet(<double(:)> value) Set the luminositiesStellar property of the disk component.

<double> massGas() Get the massGas property of the disk component.

<integer> massGasCount() Compute the count of evolvable quantities in the massGas property of the DiskVerySimple component.

<double> massGasIsGettable() Get the massGas property of the disk component.

<logical> massGasIsSettable() Specify whether the massGas property of the disk component is settable.

<void> massGasRate(<double> value) Cumulate to the rate of the massGas property of the DiskVerySimple component.

<void> massGasScale(<double> value) Set the scale of the massGas property of the DiskVerySimple component.

<void> massGasSet(<double> value) Set the massGas property of the disk component.

<double> massStellar() Get the massStellar property of the disk component.

<integer> massStellarCount() Compute the count of evolvable quantities in the massStellar property of the DiskVerySimple component.

<double> massStellarIsGettable() Get the massStellar property of the disk component.

<logical> massStellarIsSettable() Specify whether the massStellar property of the disk component is settable.

<void> massStellarRate(<double> value) Cumulate to the rate of the massStellar property of the DiskVerySimple component.

<void> massStellarScale(<double> value) Set the scale of the massStellar property of the DiskVerySimple component.

<void> massStellarSet(<double> value) Set the massStellar property of the disk component.

<void> nameFromIndex(<integer> count →, <varying_string>name ←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the disk component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.
15.5. Objects

```plaintext
<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer[:,,:]> integerBuffer ↔, <integer> doubleProperty ↔, <integer> doubleBufferCount ↔, <double[:,,:]> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

<double> radius() Get the radius property of the disk component.

<double> radiusIsGettable() Get the radius property of the disk component.

<logical> radiusIsSettable() Specify whether the radius property of the disk component is settable.

<void> radiusSet(<double> value) Set the radius property of the disk component.

<void> readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

<type(history)> starFormationHistory() Get the starFormationHistory property of the disk component.

<integer> starFormationHistoryCount() Compute the count of evolvable quantities in the starFormationHistory property of the DiskExponential component.

<type(history)> starFormationHistoryIsGettable() Get the starFormationHistory property of the disk component.

<logical> starFormationHistoryIsSettable() Specify whether the starFormationHistory property of the disk component is settable.

<void> starFormationHistoryRate(<type(history)> value) Cumulate to the rate of the starFormationHistory property of the DiskExponential component.
```
15. Coding Galacticus

```<void> starFormationHistoryScale(<type(history)> value) Set the scale of the starFormationHistory property of the DiskExponential component.```

```<void> starFormationHistorySet(<type(history)> value) Set the starFormationHistory property of the disk component.```

```<double> starFormationRate() Get the starFormationRate property of the disk component.```

```<void> starFormationRateFunction(<function()> deferredFunction) Set the function to be used for the get method of the starFormationRate property of the disk component.```

```<logical> starFormationRateIsAttached() Return whether the get method of the starFormationRate property of the disk component has been attached to a function.```

```<double> starFormationRateIsGettable() Get the starFormationRate property of the disk component.```

```<logical> starFormationRateIsSettable() Specify whether the starFormationRate property of the disk component is settable.```

```<type(history)> stellarPropertiesHistory() Get the stellarPropertiesHistory property of the disk component.```

```<integer> stellarPropertiesHistoryCount() Compute the count of evolvable quantities in the stellarPropertiesHistory property of the DiskExponential component.```

```<type(history)> stellarPropertiesHistoryIsGettable() Get the stellarPropertiesHistory property of the disk component.```

```<logical> stellarPropertiesHistoryIsSettable() Specify whether the stellarPropertiesHistory property of the disk component is settable.```

```<void> stellarPropertiesHistoryRate(<type(history)> value) Cumulate to the rate of the stellarPropertiesHistory property of the DiskExponential component.```

```<void> stellarPropertiesHistoryScale(<type(history)> value) Set the scale of the stellarPropertiesHistory property of the DiskExponential component.```

```<void> stellarPropertiesHistorySet(<type(history)> value) Set the stellarPropertiesHistory property of the disk component.```

```<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.```

```<type(varying_string)> type() Return the type of this object.```

```<double> velocity() Get the velocity property of the disk component.```

```<double> velocityIsGettable() Get the velocity property of the disk component.```

```<logical> velocityIsSettable() Specify whether the velocity property of the disk component is settable.```

```<void> velocitySet(<double> value) Set the velocity property of the disk component.```

```<logical> verySimpleIsActive() Return whether the verySimple implementation of the disk component class is active.```
15.5. Objects

nodeComponentDiskVerySimple

<type(abundances)> abundancesGas() Get the abundancesGas property of the disk component.

<integer> abundancesGasCount() Compute the count of evolvable quantities in the abundancesGas property of the DiskExponential component.

<type(abundances)> abundancesGasIsGettable() Get the abundancesGas property of the disk component.

<logical> abundancesGasIsSettable() Specify whether the abundancesGas property of the disk component is settable.

<void> abundancesGasRate(<type(abundances)> value) Cumulate to the rate of the abundancesGas property of the DiskExponential component.

<void> abundancesGasScale(<type(abundances)> value) Set the scale of the abundancesGas property of the DiskExponential component.

<void> abundancesGasSet(<type(abundances)> value) Set the abundancesGas property of the disk component.

<type(abundances)> abundancesStellar() Get the abundancesStellar property of the disk component.

<integer> abundancesStellarCount() Compute the count of evolvable quantities in the abundancesStellar property of the DiskExponential component.

<type(abundances)> abundancesStellarIsGettable() Get the abundancesStellar property of the disk component.

<logical> abundancesStellarIsSettable() Specify whether the abundancesStellar property of the disk component is settable.

<void> abundancesStellarRate(<type(abundances)> value) Cumulate to the rate of the abundancesStellar property of the DiskExponential component.

<void> abundancesStellarScale(<type(abundances)> value) Set the scale of the abundancesStellar property of the DiskExponential component.

<void> abundancesStellarSet(<type(abundances)> value) Set the abundancesStellar property of the disk component.

<double> angularMomentum() Get the angularMomentum property of the disk component.

<integer> angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the DiskExponential component.

<double> angularMomentumIsGettable() Get the angularMomentum property of the disk component.

<logical> angularMomentumIsSettable() Specify whether the angularMomentum property of the disk component is settable.

<void> angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the DiskExponential component.
15. Coding GALACTICUS

<void> angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the DiskExponential component.

<void> angularMomentumSet(<double> value) Set the angularMomentum property of the disk component.

<void> attachPipe() Attach pipes to the very simple disk component.

<void> builder(<*type(node)> componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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←) Deserializethe evolvable quantities from an array.

<void> destroy() Destroy the object.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<logical> exponentialIsActive() Return whether the exponential implementation of the disk component class is active.

<double> halfMassRadius() Get the halfMassRadius property of the disk component.

<double> halfMassRadiusIsGettable() Get the halfMassRadius property of the disk component.

<logical> halfMassRadiusIsSettable() Specify whether the halfMassRadius property of the disk component is settable.

<*type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<logical> isInitialized() Get the isInitialized property of the disk component.

<logical> isInitializedIsGettable() Get the isInitialized property of the disk component.
15.5. Objects

<logical> isInitializedIsSettable() Specify whether the isInitialized property of the disk component is settable.

<void> isInitializedSet(<logical> value) Set the isInitialized property of the disk component.

<double()> luminositiesStellar() Get the luminositiesStellar property of the disk component.

<integer> luminositiesStellarCount() Compute the count of evolvable quantities in the luminositiesStellar property of the DiskExponential component.

<double()> luminositiesStellarIsGettable() Get the luminositiesStellar property of the disk component.

<logical> luminositiesStellarIsSettable() Specify whether the luminositiesStellar property of the disk component is settable.

<void> luminositiesStellarRate(<double()> value) Cumulate to the rate of the luminositiesStellar property of the DiskExponential component.

<void> luminositiesStellarScale(<double()> value) Set the scale of the luminositiesStellar property of the DiskExponential component.

<void> luminositiesStellarSet(<double()> value) Set the luminositiesStellar property of the disk component.

<double> massGas() Get the massGas property of the disk component.

<integer> massGasCount() Compute the count of evolvable quantities in the massGas property of the DiskVerySimple component.

<double> massGasIsGettable() Get the massGas property of the disk component.

<logical> massGasIsSettable() Specify whether the massGas property of the disk component is settable.

<void> massGasRate(<double> value) Cumulate to the rate of the massGas property of the DiskVerySimple component.

<void> massGasScale(<double> value) Set the scale of the massGas property of the DiskVerySimple component.

<void> massGasSet(<double> value) Set the massGas property of the disk component.

<double> massStellar() Get the massStellar property of the disk component.

<integer> massStellarCount() Compute the count of evolvable quantities in the massStellar property of the DiskVerySimple component.

<double> massStellarIsGettable() Get the massStellar property of the disk component.

<logical> massStellarIsSettable() Specify whether the massStellar property of the disk component is settable.

<void> massStellarRate(<double> value) Cumulate to the rate of the massStellar property of the DiskVerySimple component.
massStellarScale(value) Set the scale of the massStellar property of the DiskVerySimple component.

massStellarSet(value) Set the massStellar property of the disk component.

nameFromIndex(count, name) Return the name of a property given is index.

nullIsActive() Return whether the null implementation of the disk component class is active.

odeStepRatesInitialize() Initialize rates for evolvable properties.

odeStepScalesInitialize() Initialize scales for evolvable properties.

output(integerProperty, integerBufferCount, integerBuffer, doubleProperty, doubleBufferCount, doubleBuffer, time, instance) Generate values of outputtable properties.

outputCount(integerPropertyCount, doublePropertyCount, time, instance) Compute a count of outputtable properties.

outputNames(integerProperty, integerPropertyNames, integerPropertyComments, integerPropertyUnitsSI, doubleProperty, doublePropertyNames, doublePropertyComments, doublePropertyUnitsSI, time, instance) Generate names of outputtable properties.

potential(radius, componentType, massType, haloLoaded) Compute the gravitational potential.

radius() Get the radius property of the disk component.

radiusIsGettable() Get the radius property of the disk component.

radiusIsSettable() Specify whether the radius property of the disk component is settable.

readRaw(fileHandle) Read a binary dump of the nodeComponent from the given fileHandle.

rotationCurve(radius, componentType, massType, haloLoaded) Compute the rotation curve.

rotationCurveGradient(radius, componentType, massType, haloLoaded) Compute the rotation curve gradient.

serializeCount() Return a count of the number of evolvable quantities to be evolved.

serializeRates(array) Serialize the evolvable rates to an array.

serializeScales(array) Serialize the evolvable scales to an array.

serializeValues(array) Serialize the evolvable quantities to an array.
15.5. Objects

<type(history)> starFormationHistory() Get the starFormationHistory property of the disk component.

<integer> starFormationHistoryCount() Compute the count of evolvable quantities in the starFormationHistory property of the DiskExponential component.

<type(history)> starFormationHistoryIsGettable() Get the starFormationHistory property of the disk component.

<logical> starFormationHistoryIsSettable() Specify whether the starFormationHistory property of the disk component is settable.

<void> starFormationHistoryRate(<type(history)> value) Cumulate to the rate of the starFormationHistory property of the DiskExponential component.

<void> starFormationHistoryScale(<type(history)> value) Set the scale of the starFormationHistory property of the DiskExponential component.

<void> starFormationHistorySet(<type(history)> value) Set the starFormationHistory property of the disk component.

<double> starFormationRate() Get the starFormationRate property of the disk component.

<void> starFormationRateFunction(<function()> deferredFunction) Set the function to be used for the get method of the starFormationRate property of the disk component.

<logical> starFormationRateIsAttached() Return whether the get method of the starFormationRate property of the disk component has been attached to a function.

<double> starFormationRateIsGettable() Get the starFormationRate property of the disk component.

<logical> starFormationRateIsSettable() Specify whether the starFormationRate property of the disk component is settable.

<type(history)> stellarPropertiesHistory() Get the stellarPropertiesHistory property of the disk component.

<integer> stellarPropertiesHistoryCount() Compute the count of evolvable quantities in the stellarPropertiesHistory property of the DiskExponential component.

<type(history)> stellarPropertiesHistoryIsGettable() Get the stellarPropertiesHistory property of the disk component.

<logical> stellarPropertiesHistoryIsSettable() Specify whether the stellarPropertiesHistory property of the disk component is settable.

<void> stellarPropertiesHistoryRate(<type(history)> value) Cumulate to the rate of the stellarPropertiesHistory property of the DiskExponential component.

<void> stellarPropertiesHistoryScale(<type(history)> value) Set the scale of the stellarPropertiesHistory property of the DiskExponential component.

<void> stellarPropertiesHistorySet(<type(history)> value) Set the stellarPropertiesHistory property of the disk component.
15. Coding GALACTICUS

\[
\text{\texttt{<double> surfaceDensity(<double(3)> positionCylindrical, <componentType> [componentType], <massType> [massType], <logical> [haloLoaded])}} \text{ Compute the surface density.}
\]

\[
\text{\texttt{<type(varying_string)> type()}} \text{ Return the type of this object.}
\]

\[
\text{\texttt{<double> velocity()}} \text{ Get the velocity property of the disk component.}
\]

\[
\text{\texttt{<double> velocityIsGettable()}} \text{ Get the velocity property of the disk component.}
\]

\[
\text{\texttt{<logical> velocityIsSettable()}} \text{ Specify whether the velocity property of the disk component is settable.}
\]

\[
\text{\texttt{<void> velocitySet(<double> value)}} \text{ Set the velocity property of the disk component.}
\]

\[
\text{\texttt{<logical> verySimpleIsActive()}} \text{ Return whether the verySimple implementation of the disk component class is active.}
\]

\[
\text{\texttt{<void> builder(<*type(node)> componentDefinition)}} \text{ Build a nodeComponent from a supplied XML definition.}
\]

\[
\text{\texttt{<logical> cole2000IsActive()}} \text{ Return whether the Cole2000 implementation of the formationTime component class is active.}
\]

\[
\text{\texttt{<double> density(<double(3)> positionSpherical, <componentType> [componentType], <massType> [massType], <weightBy> [weightBy], <integer> [weightIndex], <logical> [haloLoaded])}} \text{ Compute the density.}
\]

\[
\text{\texttt{<void> deserializeRates(<double(:)> array)}} \text{ Deserialize the evolvable rates from an array.}
\]

\[
\text{\texttt{<void> deserializeScales(<double(:)> array)}} \text{ Deserialize the evolvable scales from an array.}
\]

\[
\text{\texttt{<void> deserializeValues(<double(:)> array)}} \text{ Deserialize the evolvable quantities from an array.}
\]

\[
\text{\texttt{<void> destroy()}} \text{ Destroy the object.}
\]

\[
\text{\texttt{<void> dump()}} \text{ Generate an ASCII dump of all properties.}
\]

\[
\text{\texttt{<void> dumpRaw(<integer> fileHandle)}} \text{ Generate a binary dump of all properties.}
\]

\[
\text{\texttt{<double> enclosedMass(<double> radius, <componentType> [componentType], <massType> [massType], <weightBy> [weightBy], <integer> [weightIndex], <logical> [haloLoaded])}} \text{ Compute the mass enclosed within a radius.}
\]

\[
\text{\texttt{<double> formationTime()}} \text{ Get the formationTime property of the formationTime component.}
\]

\[
\text{\texttt{<double> formationTimeIsGettable()}} \text{ Get the formationTime property of the formationTime component.}
\]

\[
\text{\texttt{<logical> formationTimeIsSettable()}} \text{ Specify whether the formationTime property of the formationTime component is settable.}
\]

\[
\text{\texttt{<*type(treeNode)> host()}} \text{ Return a pointer to the host treeNode object.}
\]

\[
\text{\texttt{<void> hotHaloCoolingAbundancesRate(<type(abundances)> value)}} \text{ Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.}
\]

328
15.5. Objects

**<void> hotHaloCoolingAngularMomentumRate(<double> value)** Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

**<void> hotHaloCoolingMassRate(<double> value)** Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

**<void> initialize()** Initialize the object.

**<void> nameFromIndex(<integer> count →, <varying_string>name←)** Return the name of a property given is index.

**<logical> nullIsActive()** Return whether the null implementation of the formationTime component class is active.

**<void> odeStepRatesInitialize()** Initialize rates for evolvable properties.

**<void> odeStepScalesInitialize()** Initialize scales for evolvable properties.

**<void> output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer[:,:] integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double[:,:] doubleBuffer↔, <double> time→, <integer> instance→)** Generate values of outputtable properties.

**<void> outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→)** Compute a count of outputtable properties.

**<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.

**<double> potential(<double> radius→, <componentType [componentType]→, <massType [massType]→, <logical> [haloLoaded]→)** Compute the gravitational potential.

**<double> rotationCurve(<double> radius→, <componentType [componentType]→, <massType [massType]→, <logical> [haloLoaded]→)** Compute the rotation curve.

**<double> rotationCurveGradient(<double> radius→, <componentType [componentType]→, <massType [massType]→, <logical> [haloLoaded]→)** Compute the rotation curve gradient.

**<integer> serializeCount()** Return a count of the number of evolvable quantities to be evolved.

**<void> serializeRates(<double[:] array→)** Serialize the evolvable rates to an array.

**<void> serializeScales(<double[:] array→)** Serialize the evolvable scales to an array.

**<void> serializeValues(<double[:] array→)** Serialize the evolvable quantities to an array.

**<double> surfaceDensity(<double> positionCylindrical→, <componentType [componentType]→, <massType [massType]→, <logical> [haloLoaded]→)** Compute the surface density.

**<type(varying_string)> type()** Return the type of this object.
nodeComponentFormationTimeCole2000

<void> builder(<*type(node)> componentDefinition→) Build a nodeComponent from a supplied XML definition.

<logical> cole2000IsActive() Return whether the Cole2000 implementation of the formationTime component class is active.

<double> 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←) Deserializethe evolvable quantities from an array.

<void> destroy() Destroy the object.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<double> formationTime() Get the formationTime property of the formationTime component.

<double> formationTimeIsGettable() Get the formationTime property of the formationTime component.

<logical> formationTimeIsSettable() Specify whether the formationTime property of the formationTime component is settable.

<*type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the formationTime component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.
15.5. Objects

```c
<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)>
integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)>
doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of out-puttable properties.

<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double>
time →, <integer> instance →) Compute a count of out-puttable properties.

<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 out-
puttable properties.

<double> potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →,
<logical> [haloLoaded] →) Compute the gravitational potential.

<void> readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given
fileHandle.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType>
[massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType>
[massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

<double> surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →,
<massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

nodeComponentFormationTimeNull

<void> builder(<*type(node)>componentDefinition →) Build a nodeComponent from a supplied
XML definition.

<logical> cole2000IsActive() Return whether the Cole2000 implementation of the formationTime
class is active.

<double> 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.
```
15. Coding GALACTICUS

```c
<void> destroy() Destroy the object.
<void> dump() Generate an ASCII dump of all properties.
<void> dumpRaw(<integer> fileHandle) Generate a binary dump of all properties.
<double> enclosedMass(<double> radius, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.
<double> formationTime() Get the formationTime property of the formationTime component.
<double> formationTimeIsGettable() Get the formationTime property of the formationTime component.
<logical> formationTimeIsSettable() Specify whether the formationTime property of the formationTime component is settable.
<type(treeNode)> host() Return a pointer to the host treeNode object.
<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.
<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.
<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.
<void> initialize() Initialize the object.
<void> nameFromIndex(<integer> count, <varying_string>name) Return the name of a property given is index.
<logical> nullIsActive() Return whether the null implementation of the formationTime component class is active.
<void> odeStepRatesInitialize() Initialize rates for evolvable properties.
<void> odeStepScalesInitialize() Initialize scales for evolvable properties.
<void> output(<integer> integerProperty, <integer> integerBufferCount, <integer(:,:) > integerBuffer→, <integer>doubleProperty, <integer> doubleBufferCount, <double(:,:) > doubleBuffer→, <double> time→, <integer> instance→) Generate values of outputtable properties.
<void> outputCount(<integer> integerPropertyCount, <integer> doublePropertyCount, <double> time→, <integer> instance→) Compute a count of outputtable properties.
<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.
<double> potential(<double> radius, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.
```
15.5. Objects

<void> readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

<double> surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

nodeComponentHotHalo

<type(abundances)> abundances() Get the abundances property of the hotHalo component.

<integer> abundancesCount() Compute the count of evolvable quantities in the abundances property of the HotHaloStandard component.

<type(abundances)> abundancesIsGettable() Get the abundances property of the hotHalo component.

<logical> abundancesIsSettable() Specify whether the abundances property of the hotHalo component is settable.

<void> abundancesRate(<type(abundances)> value) Cumulate to the rate of the abundances property of the HotHaloStandard component.

<void> abundancesScale(<type(abundances)> value) Set the scale of the abundances property of the HotHaloStandard component.

<void> abundancesSet(<type(abundances)> value) Set the abundances property of the hotHalo component.

<double> angularMomentum() Get the angularMomentum property of the hotHalo component.

<integer> angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the HotHaloStandard component.

<double> angularMomentumIsGettable() Get the angularMomentum property of the hotHalo component.

<logical> angularMomentumIsSettable() Specify whether the angularMomentum property of the hotHalo component is settable.

<void> angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the HotHaloStandard component.
**15. Coding GALACTICUS**

```c
<void> angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the HotHaloStandard component.

<void> angularMomentumSet(<double> value) Set the angularMomentum property of the hotHalo component.

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<type(chemicalAbundances)> chemicals() Get the chemicals property of the hotHalo component.

<integer> chemicalsCount() Compute the count of evolvable quantities in the chemicals property of the HotHaloStandard component.

<type(chemicalAbundances)> chemicalsIsGettable() Get the chemicals property of the hotHalo component.

<logical> 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.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<double> heatSource() Get the heatSource property of the hotHalo component.

<integer> heatSourceCount() Compute the count of evolvable quantities in the heatSource property of the HotHaloStandard component.

<double> heatSourceIsGettable() Get the heatSource property of the hotHalo component.
```
15.5. Objects

<logical> heatSourceIsSettable() Specify whether the heatSource property of the hotHalo component is settable.

<void> heatSourceRate(<double> value) Cumulate to the rate of the heatSource property of the HotHaloStandard component.

<void> heatSourceRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the heatSource property of the hotHalo component.

<logical> heatSourceRateIsAttached() Return whether the rate method of the heatSource property of the hotHalo component has been attached to a function.

<type(treeNode)> host() Return a pointer to the host treeNode object.

<type(abundances)> hotHaloCoolingAbundances() Get the hotHaloCoolingAbundances property of the hotHalo component.

<integer> hotHaloCoolingAbundancesCount() Compute the count of evolvable quantities in the hotHaloCoolingAbundances property of the HotHaloStandard component.

<type(abundances)> hotHaloCoolingAbundancesIsGettable() Get the hotHaloCoolingAbundances property of the hotHalo component.

<logical> hotHaloCoolingAbundancesIsSettable() Specify whether the hotHaloCoolingAbundances property of the hotHalo component is settable.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAbundancesRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAbundances property of the hotHalo component.

<logical> hotHaloCoolingAbundancesRateIsAttached() Return whether the rate method of the hotHaloCoolingAbundances property of the hotHalo component has been attached to a function.

<double> hotHaloCoolingAngularMomentum() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

<integer> hotHaloCoolingAngularMomentumCount() Compute the count of evolvable quantities in the hotHaloCoolingAngularMomentum property of the HotHaloStandard component.

<double> hotHaloCoolingAngularMomentumIsGettable() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

<logical> hotHaloCoolingAngularMomentumIsSettable() Specify whether the hotHaloCoolingAngularMomentum property of the hotHalo component is settable.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component.
15. Coding GALACTICUS

<logical> hotHaloCoolingAngularMomentumRateIsAttached() Return whether the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component has been attached to a function.

<double> hotHaloCoolingMass() Get the hotHaloCoolingMass property of the hotHalo component.

<integer> hotHaloCoolingMassCount() Compute the count of evolvable quantities in the hotHaloCoolingMass property of the HotHaloStandard component.

<double> hotHaloCoolingMassIsGettable() Get the hotHaloCoolingMass property of the hotHalo component.

<logical> hotHaloCoolingMassIsSettable() Specify whether the hotHaloCoolingMass property of the hotHalo component is settable.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> hotHaloCoolingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingMass property of the hotHalo component.

<logical> hotHaloCoolingMassRateIsAttached() Return whether the rate method of the hotHaloCoolingMass property of the hotHalo component has been attached to a function.

<void> initialize() Initialize the object.

<double> mass() Get the mass property of the hotHalo component.

<integer> massCount() Compute the count of evolvable quantities in the mass property of the HotHaloStandard component.

<double> massIsGettable() Get the mass property of the hotHalo component.

<logical> massIsSettable() Specify whether the mass property of the hotHalo component is settable.

<void> massRate(<double> value) Cumulate to the rate of the mass property of the HotHaloStandard component.

<void> massScale(<double> value) Set the scale of the mass property of the HotHaloStandard component.

<void> massSet(<double> value) Set the mass property of the hotHalo component.

<double> massSink() Get the massSink property of the hotHalo component.

<double> massSinkIsGettable() Get the massSink property of the hotHalo component.

<logical> massSinkIsSettable() Specify whether the massSink property of the hotHalo component is settable.

<void> massSinkSet(<double> value) Set the massSink property of the hotHalo component.

<void> nameFromIndex(<integer> count →, <varying_string>name←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the hotHalo component class is active.
15.5. Objects

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<double> outerRadius() Get the outerRadius property of the hotHalo component.

<integer> outerRadiusCount() Compute the count of evolvable quantities in the outerRadius property of the HotHaloStandard component.

<void> outerRadiusFunction(<function()> deferredFunction) Set the function to be used for the get method of the outerRadius property of the hotHalo component.

<logical> outerRadiusIsAttached() Return whether the get method of the outerRadius property of the hotHalo component has been attached to a function.

<double> outerRadiusIsGettable() Get the outerRadius property of the hotHalo component.

<logical> outerRadiusIsSettable() Specify whether the outerRadius property of the hotHalo component is settable.

<void> outerRadiusRate(<double> value) Cumulate to the rate of the outerRadius property of the HotHaloStandard component.

<void> outerRadiusScale(<double> value) Set the scale of the outerRadius property of the HotHaloStandard component.

<void> outerRadiusSet(<double> value) Set the outerRadius property of the hotHalo component.

<type(abundances)> outflowedAbundances() Get the outflowedAbundances property of the hotHalo component.

<integer> outflowedAbundancesCount() Compute the count of evolvable quantities in the outflowedAbundances property of the HotHaloStandard component.

<type(abundances)> outflowedAbundancesIsGettable() Get the outflowedAbundances property of the hotHalo component.

<logical> outflowedAbundancesIsSettable() Specify whether the outflowedAbundances property of the hotHalo component is settable.

<void> outflowedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the outflowedAbundances property of the HotHaloStandard component.

<void> outflowedAbundancesScale(<type(abundances)> value) Set the scale of the outflowedAbundances property of the HotHaloStandard component.

<void> outflowedAbundancesSet(<type(abundances)> value) Set the outflowedAbundances property of the hotHalo component.

<double> outflowedAngularMomentum() Get the outflowedAngularMomentum property of the hotHalo component.

<integer> outflowedAngularMomentumCount() Compute the count of evolvable quantities in the outflowedAngularMomentum property of the HotHaloStandard component.

<double> outflowedAngularMomentumIsGettable() Get the outflowedAngularMomentum property of the hotHalo component.
15. Coding GALACTICUS

<logical> outflowedAngularMomentumIsSettable() Specify whether the outflowedAngularMomentum property of the hotHalo component is settable.

<void> outflowedAngularMomentumRate(<double> value) Cumulate to the rate of the outflowedAngularMomentum property of the HotHaloStandard component.

<void> outflowedAngularMomentumScale(<double> value) Set the scale of the outflowedAngularMomentum property of the HotHaloStandard component.

<void> outflowedAngularMomentumSet(<double> value) Set the outflowedAngularMomentum property of the hotHalo component.

<double> outflowedMass() Get the outflowedMass property of the hotHalo component.

<integer> outflowedMassCount() Compute the count of evolvable quantities in the outflowedMass property of the HotHaloStandard component.

<double> outflowedMassIsGettable() Get the outflowedMass property of the hotHalo component.

<logical> 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.

<type(abundances)> outflowingAbundances() Get the outflowingAbundances property of the hotHalo component.

<integer> outflowingAbundancesCount() Compute the count of evolvable quantities in the outflowingAbundances property of the HotHaloStandard component.

<type(abundances)> outflowingAbundancesIsGettable() Get the outflowingAbundances property of the hotHalo component.

<logical> 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.

<void> outflowingAbundancesRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingAbundances property of the hotHalo component.

<logical> outflowingAbundancesRateIsAttached() Return whether the rate method of the outflowingAbundances property of the hotHalo component has been attached to a function.

<double> outflowingAngularMomentum() Get the outflowingAngularMomentum property of the hotHalo component.

<integer> outflowingAngularMomentumCount() Compute the count of evolvable quantities in the outflowingAngularMomentum property of the HotHaloStandard component.
15.5. Objects

<double> outflowingAngularMomentumIsGettable() Get the outflowingAngularMomentum property of the hotHalo component.

<logical> outflowingAngularMomentumIsSettable() Specify whether the outflowingAngularMomentum property of the hotHalo component is settable.

<void> outflowingAngularMomentumRate(<double> value) Cumulate to the rate of the outflowingAngularMomentum property of the HotHaloStandard component.

<void> outflowingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingAngularMomentum property of the hotHalo component.

<logical> outflowingAngularMomentumRateIsAttached() Return whether the rate method of the outflowingAngularMomentum property of the hotHalo component has been attached to a function.

<double> outflowingMass() Get the outflowingMass property of the hotHalo component.

<integer> outflowingMassCount() Compute the count of evolvable quantities in the outflowingMass property of the HotHaloStandard component.

<double> outflowingMassIsGettable() Get the outflowingMass property of the hotHalo component.

<logical> outflowingMassIsSettable() Specify whether the outflowingMass property of the hotHalo component is settable.

<void> outflowingMassRate(<double> value) Cumulate to the rate of the outflowingMass property of the HotHaloStandard component.

<void> outflowingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingMass property of the hotHalo component.

<logical> outflowingMassRateIsAttached() Return whether the rate method of the outflowingMass property of the hotHalo component has been attached to a function.

<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.
15. Coding GALACTICUS

```c
<double> rotationCurveGradient(<double> radius →, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the hotHalo component class is active.

<type(abundances)> strippedAbundances() Get the strippedAbundances property of the hotHalo component.

<integer> strippedAbundancesCount() Compute the count of evolvable quantities in the strippedAbundances property of the HotHaloStandard component.

<type(abundances)> strippedAbundancesIsGettable() Get the strippedAbundances property of the hotHalo component.

<logical> strippedAbundancesIsSettable() Specify whether the strippedAbundances property of the hotHalo component is settable.

<void> strippedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the strippedAbundances property of the HotHaloStandard component.

<void> strippedAbundancesScale(<type(abundances)> value) Set the scale of the strippedAbundances property of the HotHaloStandard component.

<void> strippedAbundancesSet(<type(abundances)> value) Set the strippedAbundances property of the hotHalo component.

<double> strippedMass() Get the strippedMass property of the hotHalo component.

<integer> strippedMassCount() Compute the count of evolvable quantities in the strippedMass property of the HotHaloStandard component.

<double> strippedMassIsGettable() Get the strippedMass property of the hotHalo component.

<logical> strippedMassIsSettable() Specify whether the strippedMass property of the hotHalo component is settable.

<void> strippedMassRate(<double> value) Cumulate to the rate of the strippedMass property of the HotHaloStandard component.

<void> strippedMassScale(<double> value) Set the scale of the strippedMass property of the HotHaloStandard component.

<void> strippedMassSet(<double> value) Set the strippedMass property of the hotHalo component.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.
```
15.5. Objects

\[\text{type(\text{varying\_string})} \] type() Return the type of this object.

\[\text{double} \] unaccretedMass() Get the unaccretedMass property of the hotHalo component.

\[\text{integer} \] unaccretedMassCount() Compute the count of evolvable quantities in the unaccretedMass property of the HotHaloStandard component.

\[\text{double} \] unaccretedMassIsGettable() Get the unaccretedMass property of the hotHalo component.

\[\text{logical} \] unaccretedMassIsSettable() Specify whether the unaccretedMass property of the hotHalo component is settable.

\[\text{void} \] unaccretedMassRate(\[\text{double} \] value) Cumulate to the rate of the unaccretedMass property of the HotHaloStandard component.

\[\text{void} \] unaccretedMassScale(\[\text{double} \] value) Set the scale of the unaccretedMass property of the HotHaloStandard component.

\[\text{void} \] unaccretedMassSet(\[\text{double} \] value) Set the unaccretedMass property of the hotHalo component.

\[\text{logical} \] verySimpleIsActive() Return whether the verySimple implementation of the hotHalo component class is active.

\text{nodeComponentHotHaloNull}

\[\text{type(abundances)} \] abundances() Get the abundances property of the hotHalo component.

\[\text{integer} \] abundancesCount() Compute the count of evolvable quantities in the abundances property of the HotHaloStandard component.

\[\text{type(abundances)} \] abundancesIsGettable() Get the abundances property of the hotHalo component.

\[\text{logical} \] abundancesIsSettable() Specify whether the abundances property of the hotHalo component is settable.

\[\text{void} \] abundancesRate(\[\text{type(abundances)} \] value) Cumulate to the rate of the abundances property of the HotHaloStandard component.

\[\text{void} \] abundancesScale(\[\text{type(abundances)} \] value) Set the scale of the abundances property of the HotHaloStandard component.

\[\text{void} \] abundancesSet(\[\text{type(abundances)} \] value) Set the abundances property of the hotHalo component.

\[\text{double} \] angularMomentum() Get the angularMomentum property of the hotHalo component.

\[\text{integer} \] angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the HotHaloStandard component.

\[\text{double} \] angularMomentumIsGettable() Get the angularMomentum property of the hotHalo component.
15. Coding GALACTICUS

<logical> angularMomentumIsSettable() Specify whether the angularMomentum property of the hotHalo component is settable.

<void> angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the HotHaloStandard component.

<void> angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the HotHaloStandard component.

<void> angularMomentumSet(<double> value) Set the angularMomentum property of the hotHalo component.

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<type(chemicalAbundances)> chemicals() Get the chemicals property of the hotHalo component.

<integer> chemicalsCount() Compute the count of evolvable quantities in the chemicals property of the HotHaloStandard component.

<type(chemicalAbundances)> chemicalsIsGettable() Get the chemicals property of the hotHalo component.

<logical> 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.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<double> heatSource() Get the heatSource property of the hotHalo component.
15.5. Objects

<integer> heatSourceCount() Compute the count of evolvable quantities in the heatSource property of the HotHaloStandard component.

<double> heatSourceIsGettable() Get the heatSource property of the hotHalo component.

<logical> heatSourceIsSettable() Specify whether the heatSource property of the hotHalo component is settable.

<void> heatSourceRate(<double> value) Cumulate to the rate of the heatSource property of the HotHaloStandard component.

<void> heatSourceRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the heatSource property of the hotHalo component.

<logical> heatSourceRateIsAttached() Return whether the rate method of the heatSource property of the hotHalo component has been attached to a function.

<*type(treeNode)> host() Return a pointer to the host treeNode object.

<type(abundances)> hotHaloCoolingAbundances() Get the hotHaloCoolingAbundances property of the hotHalo component.

<integer> hotHaloCoolingAbundancesCount() Compute the count of evolvable quantities in the hotHaloCoolingAbundances property of the HotHaloStandard component.

<type(abundances)> hotHaloCoolingAbundancesIsGettable() Get the hotHaloCoolingAbundances property of the hotHalo component.

<logical> hotHaloCoolingAbundancesIsSettable() Specify whether the hotHaloCoolingAbundances property of the hotHalo component is settable.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAbundancesRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAbundances property of the hotHalo component.

<logical> hotHaloCoolingAbundancesRateIsAttached() Return whether the rate method of the hotHaloCoolingAbundances property of the hotHalo component has been attached to a function.

<double> hotHaloCoolingAngularMomentum() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

<integer> hotHaloCoolingAngularMomentumCount() Compute the count of evolvable quantities in the hotHaloCoolingAngularMomentum property of the HotHaloStandard component.

<double> hotHaloCoolingAngularMomentumIsGettable() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

<logical> hotHaloCoolingAngularMomentumIsSettable() Specify whether the hotHaloCoolingAngularMomentum property of the hotHalo component is settable.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.
15. Coding GALACTICUS

`<void>` hotHaloCoolingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component.

`<logical>` hotHaloCoolingAngularMomentumRateIsAttached() Return whether the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component has been attached to a function.

`<double>` hotHaloCoolingMass() Get the hotHaloCoolingMass property of the hotHalo component.

`<integer>` hotHaloCoolingMassCount() Compute the count of evolvable quantities in the hotHaloCoolingMass property of the HotHaloStandard component.

`<double>` hotHaloCoolingMassIsGettable() Get the hotHaloCoolingMass property of the hotHalo component.

`<logical>` hotHaloCoolingMassIsSettable() Specify whether the hotHaloCoolingMass property of the hotHalo component is settable.

`<void>` hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

`<void>` hotHaloCoolingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingMass property of the hotHalo component.

`<logical>` hotHaloCoolingMassRateIsAttached() Return whether the rate method of the hotHaloCoolingMass property of the hotHalo component has been attached to a function.

`<void>` initialize() Initialize the object.

`<double>` mass() Get the mass property of the hotHalo component.

`<integer>` massCount() Compute the count of evolvable quantities in the mass property of the HotHaloStandard component.

`<double>` massIsGettable() Get the mass property of the hotHalo component.

`<logical>` massIsSettable() Specify whether the mass property of the hotHalo component is settable.

`<void>` massRate(<double> value) Cumulate to the rate of the mass property of the HotHaloStandard component.

`<void>` massScale(<double> value) Set the scale of the mass property of the HotHaloStandard component.

`<void>` massSet(<double> value) Set the mass property of the hotHalo component.

`<double>` massSink() Get the massSink property of the hotHalo component.

`<double>` massSinkIsGettable() Get the massSink property of the hotHalo component.

`<logical>` massSinkIsSettable() Specify whether the massSink property of the hotHalo component is settable.

`<void>` massSinkSet(<double> value) Set the massSink property of the hotHalo component.
15.5. Objects

<void> nameFromIndex(<integer> count →, <varying_string>name←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the hotHalo component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<double> outerRadius() Get the outerRadius property of the hotHalo component.

<integer> outerRadiusCount() Compute the count of evolvable quantities in the outerRadius property of the HotHaloStandard component.

<void> outerRadiusFunction(<function()> deferredFunction) Set the function to be used for the get method of the outerRadius property of the hotHalo component.

<logical> outerRadiusIsAttached() Return whether the get method of the outerRadius property of the hotHalo component has been attached to a function.

<double> outerRadiusIsGettable() Get the outerRadius property of the hotHalo component.

<logical> outerRadiusIsSettable() Specify whether the outerRadius property of the hotHalo component is settable.

<void> outerRadiusRate(<double> value) Cumulate to the rate of the outerRadius property of the HotHaloStandard component.

<void> outerRadiusScale(<double> value) Set the scale of the outerRadius property of the HotHaloStandard component.

<void> outerRadiusSet(<double> value) Set the outerRadius property of the hotHalo component.

<type(abundances)> outflowedAbundances() Get the outflowedAbundances property of the hotHalo component.

<integer> outflowedAbundancesCount() Compute the count of evolvable quantities in the outflowedAbundances property of the HotHaloStandard component.

<type(abundances)> outflowedAbundancesIsGettable() Get the outflowedAbundances property of the hotHalo component.

<logical> outflowedAbundancesIsSettable() Specify whether the outflowedAbundances property of the hotHalo component is settable.

<void> outflowedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the outflowedAbundances property of the HotHaloStandard component.

<void> outflowedAbundancesScale(<type(abundances)> value) Set the scale of the outflowedAbundances property of the HotHaloStandard component.

<void> outflowedAbundancesSet(<type(abundances)> value) Set the outflowedAbundances property of the hotHalo component.

<double> outflowedAngularMomentum() Get the outflowedAngularMomentum property of the hotHalo component.
15. Coding GALACTICUS

`<integer> outflowedAngularMomentumCount()` Compute the count of evolvable quantities in the `outflowedAngularMomentum` property of the `HotHaloStandard` component.

`<double> outflowedAngularMomentumIsGettable()` Get the `outflowedAngularMomentum` property of the `hotHalo` component.

`<logical> outflowedAngularMomentumIsSettable()` Specify whether the `outflowedAngularMomentum` property of the `HotHaloStandard` component is settable.

`<void> outflowedAngularMomentumRate(<double> value)` Cumulate to the rate of the `outflowedAngularMomentum` property of the `HotHaloStandard` component.

`<void> outflowedAngularMomentumScale(<double> value)` Set the scale of the `outflowedAngularMomentum` property of the `HotHaloStandard` component.

`<void> outflowedAngularMomentumSet(<double> value)` Set the `outflowedAngularMomentum` property of the `hotHalo` component.

`<double> outflowedMass()` Get the `outflowedMass` property of the `hotHalo` component.

`<integer> outflowedMassCount()` Compute the count of evolvable quantities in the `outflowedMass` property of the `HotHaloStandard` component.

`<double> outflowedMassIsGettable()` Get the `outflowedMass` property of the `hotHalo` component.

`<logical> 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.

`<type(abundances)> outflowingAbundances()` Get the `outflowingAbundances` property of the `hotHalo` component.

`<integer> outflowingAbundancesCount()` Compute the count of evolvable quantities in the `outflowingAbundances` property of the `HotHaloStandard` component.

`<type(abundances)> outflowingAbundancesIsGettable()` Get the `outflowingAbundances` property of the `hotHalo` component.

`<logical> 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.

`<void> outflowingAbundancesRateFunction(<function()> deferredFunction)` Set the function to be used for the `rate` method of the `outflowingAbundances` property of the `hotHalo` component.

`<logical> outflowingAbundancesRateIsAttached()` Return whether the rate method of the `outflowingAbundances` property of the `hotHalo` component has been attached to a function.
15.5. Objects

<double> outflowingAngularMomentum() Get the outflowingAngularMomentum property of the hotHalo component.

<integer> outflowingAngularMomentumCount() Compute the count of evolvable quantities in the outflowingAngularMomentum property of the HotHaloStandard component.

<double> outflowingAngularMomentumIsGettable() Get the outflowingAngularMomentum property of the hotHalo component.

<logical> outflowingAngularMomentumIsSettable() Specify whether the outflowingAngularMomentum property of the hotHalo component is settable.

<void> outflowingAngularMomentumRate(<double> value) Cumulate to the rate of the outflowingAngularMomentum property of the HotHaloStandard component.

<void> outflowingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingAngularMomentum property of the hotHalo component.

<logical> outflowingAngularMomentumRateIsAttached() Return whether the rate method of the outflowingAngularMomentum property of the hotHalo component has been attached to a function.

<double> outflowingMass() Get the outflowingMass property of the hotHalo component.

<integer> outflowingMassCount() Compute the count of evolvable quantities in the outflowingMass property of the HotHaloStandard component.

<double> outflowingMassIsGettable() Get the outflowingMass property of the hotHalo component.

<logical> outflowingMassIsSettable() Specify whether the outflowingMass property of the hotHalo component is settable.

<void> outflowingMassRate(<double> value) Cumulate to the rate of the outflowingMass property of the HotHaloStandard component.

<void> outflowingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingMass property of the hotHalo component.

<logical> outflowingMassRateIsAttached() Return whether the rate method of the outflowingMass property of the hotHalo component has been attached to a function.

<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:, :) > integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:, :) > doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

<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.
15. Coding GALACTICUS

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the hotHalo component class is active.

<type(abundances)> strippedAbundances() Get the strippedAbundances property of the hotHalo component.

<integer> strippedAbundancesCount() Compute the count of evolvable quantities in the strippedAbundances property of the HotHaloStandard component.

<type(abundances)> strippedAbundancesIsGettable() Get the strippedAbundances property of the hotHalo component.

<logical> strippedAbundancesIsSettable() Specify whether the strippedAbundances property of the hotHalo component is settable.

<void> strippedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the strippedAbundances property of the HotHaloStandard component.

<void> strippedAbundancesScale(<type(abundances)> value) Set the scale of the strippedAbundances property of the HotHaloStandard component.

<void> strippedAbundancesSet(<type(abundances)> value) Set the strippedAbundances property of the hotHalo component.

<double> strippedMass() Get the strippedMass property of the hotHalo component.

<integer> strippedMassCount() Compute the count of evolvable quantities in the strippedMass property of the HotHaloStandard component.

<double> strippedMassIsGettable() Get the strippedMass property of the hotHalo component.

<logical> strippedMassIsSettable() Specify whether the strippedMass property of the hotHalo component is settable.

<void> strippedMassRate(<double> value) Cumulate to the rate of the strippedMass property of the HotHaloStandard component.
15.5. Objects

<void> strippedMassScale(<double> value) Set the scale of the strippedMass property of the HotHaloStandard component.

<void> strippedMassSet(<double> value) Set the strippedMass property of the hotHalo component.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

<double> unaccretedMass() Get the unaccretedMass property of the hotHalo component.

<integer> unaccretedMassCount() Compute the count of evolvable quantities in the unaccretedMass property of the HotHaloStandard component.

<double> unaccretedMassIsGettable() Get the unaccretedMass property of the hotHalo component.

<logical> unaccretedMassIsSettable() Specify whether the unaccretedMass property of the hotHalo component is settable.

<void> unaccretedMassRate(<double> value) Cumulate to the rate of the unaccretedMass property of the HotHaloStandard component.

<void> unaccretedMassScale(<double> value) Set the scale of the unaccretedMass property of the HotHaloStandard component.

<void> unaccretedMassSet(<double> value) Set the unaccretedMass property of the hotHalo component.

<logical> verySimpleIsActive() Return whether the verySimple implementation of the hotHalo component class is active.

nodeComponentHotHaloStandard

<type(abundances)> abundances() Get the abundances property of the hotHalo component.

<integer> abundancesCount() Compute the count of evolvable quantities in the abundances property of the HotHaloStandard component.

<type(abundances)> abundancesIsGettable() Get the abundances property of the hotHalo component.

<logical> abundancesIsSettable() Specify whether the abundances property of the hotHalo component is settable.

<void> abundancesRate(<type(abundances)> value) Cumulate to the rate of the abundances property of the HotHaloStandard component.

<void> abundancesScale(<type(abundances)> value) Set the scale of the abundances property of the HotHaloStandard component.

<void> abundancesSet(<type(abundances)> value) Set the abundances property of the hotHalo component.
15. Coding GALACTICUS

```plaintext
<double> angularMomentum() Get the angularMomentum property of the hotHalo component.

<integer> angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the HotHaloStandard component.

<double> angularMomentumIsGettable() Get the angularMomentum property of the hotHalo component.

<logical> angularMomentumIsSettable() Specify whether the angularMomentum property of the hotHalo component is settable.

<void> angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the HotHaloStandard component.

<void> angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the HotHaloStandard component.

<void> angularMomentumSet(<double> value) Set the angularMomentum property of the hotHalo component.

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<type(chemicalAbundances)> chemicals() Get the chemicals property of the hotHalo component.

<integer> chemicalsCount() Compute the count of evolvable quantities in the chemicals property of the HotHaloStandard component.

<type(chemicalAbundances)> chemicalsIsGettable() Get the chemicals property of the hotHalo component.

<logical> 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.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.
```
15.5. Objects

```c
<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)
Compute the mass enclosed within a radius.

<double> heatSource() Get the heatSource property of the hotHalo component.

<integer> heatSourceCount() Compute the count of evolvable quantities in the heatSource property of the HotHaloStandard component.

<double> heatSourceIsGettable() Get the heatSource property of the hotHalo component.

<logical> heatSourceIsSettable() Specify whether the heatSource property of the hotHalo component is settable.

<void> heatSourceRate(<double> value) Cumulate to the rate of the heatSource property of the hotHalo component.

<void> heatSourceRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the heatSource property of the hotHalo component.

<logical> heatSourceRateIsAttached() Return whether the rate method of the heatSource property of the hotHalo component has been attached to a function.

<*>host() Return a pointer to the host treeNode object.

<type(abundances)> hotHaloCoolingAbundances() Get the hotHaloCoolingAbundances property of the hotHalo component.

<integer> hotHaloCoolingAbundancesCount() Compute the count of evolvable quantities in the hotHaloCoolingAbundances property of the HotHaloStandard component.

<type(abundances)> hotHaloCoolingAbundancesIsGettable() Get the hotHaloCoolingAbundances property of the hotHalo component.

<logical> hotHaloCoolingAbundancesIsSettable() Specify whether the hotHaloCoolingAbundances property of the hotHalo component is settable.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAbundancesRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAbundances property of the hotHalo component.

<logical> hotHaloCoolingAbundancesRateIsAttached() Return whether the rate method of the hotHaloCoolingAbundances property of the hotHalo component has been attached to a function.

<double> hotHaloCoolingAngularMomentum() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

<integer> hotHaloCoolingAngularMomentumCount() Compute the count of evolvable quantities in the hotHaloCoolingAngularMomentum property of the HotHaloStandard component.

<double> hotHaloCoolingAngularMomentumIsGettable() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.
```
<logical> hotHaloCoolingAngularMomentumIsSettable() Specify whether the hotHaloCoolingAngularMomentum property of the hotHalo component is settable.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<logical> hotHaloCoolingAngularMomentumRateIsAttached() Return whether the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component has been attached to a function.

<double> hotHaloCoolingMass() Get the hotHaloCoolingMass property of the hotHalo component.

<integer> hotHaloCoolingMassCount() Compute the count of evolvable quantities in the hotHaloCoolingMass property of the HotHaloStandard component.

<double> hotHaloCoolingMassIsGettable() Get the hotHaloCoolingMass property of the hotHalo component.

<logical> hotHaloCoolingMassIsSettable() Specify whether the hotHaloCoolingMass property of the hotHalo component is settable.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> hotHaloCoolingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingMass property of the hotHalo component.

<logical> hotHaloCoolingMassRateIsAttached() Return whether the rate method of the hotHaloCoolingMass property of the hotHalo component has been attached to a function.

<void> initialize() Initialize the object.

<double> mass() Get the mass property of the hotHalo component.

<integer> massCount() Compute the count of evolvable quantities in the mass property of the HotHaloStandard component.

<double> massIsGettable() Get the mass property of the hotHalo component.

<logical> massIsSettable() Specify whether the mass property of the hotHalo component is settable.

<void> massRate(<double> value) Cumulate to the rate of the mass property of the HotHaloStandard component.

<void> massScale(<double> value) Set the scale of the mass property of the HotHaloStandard component.

<void> massSet(<double> value) Set the mass property of the hotHalo component.

<double> massSink() Get the massSink property of the hotHalo component.

<double> massSinkIsGettable() Get the massSink property of the hotHalo component.
15.5. Objects

**logical** massSinkIsSettable() Specify whether the massSink property of the hotHalo component is settable.

**void** massSinkSet(<double> value) Set the massSink property of the hotHalo component.

**void** nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

**logical** nullIsActive() Return whether the null implementation of the hotHalo component class is active.

**void** odeStepRatesInitialize() Initialize rates for evolvable properties.

**void** odeStepScalesInitialize() Initialize scales for evolvable properties.

**double** outerRadius() Get the outerRadius property of the hotHalo component.

**integer** outerRadiusCount() Compute the count of evolvable quantities in the outerRadius property of the HotHaloStandard component.

**void** outerRadiusFunction(<function()> deferredFunction) Set the function to be used for the get method of the outerRadius property of the hotHalo component.

**logical** outerRadiusIsAttached() Return whether the get method of the outerRadius property of the hotHalo component has been attached to a function.

**double** outerRadiusIsGettable() Get the outerRadius property of the hotHalo component.

**logical** outerRadiusIsSettable() Specify whether the outerRadius property of the hotHalo component is settable.

**void** outerRadiusRate(<double> value) Cumulate to the rate of the outerRadius property of the HotHaloStandard component.

**void** outerRadiusScale(<double> value) Set the scale of the outerRadius property of the HotHaloStandard component.

**void** outerRadiusSet(<double> value) Set the outerRadius property of the hotHalo component.

**double** outerRadiusValue() Get the outerRadius property of the hotHalo component.

**type(abundances)> outflowedAbundances() Get the outflowedAbundances property of the hotHalo component.

**integer> outflowedAbundancesCount() Compute the count of evolvable quantities in the outflowedAbundances property of the HotHaloStandard component.

**type(abundances)> outflowedAbundancesIsGettable() Get the outflowedAbundances property of the hotHalo component.

**logical> outflowedAbundancesIsSettable() Specify whether the outflowedAbundances property of the hotHalo component is settable.

**void> outflowedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the outflowedAbundances property of the HotHaloStandard component.
15. Coding Galacticus

```c
<void> outflowedAbundancesScale(<type(abundances)> value) Set the scale of the outflowedAbundances property of the HotHaloStandard component.

<void> outflowedAbundancesSet(<type(abundances)> value) Set the outflowedAbundances property of the hotHalo component.

<double> outflowedAngularMomentum() Get the outflowedAngularMomentum property of the hotHalo component.

<integer> outflowedAngularMomentumCount() Compute the count of evolvable quantities in the outflowedAngularMomentum property of the HotHaloStandard component.

<double> outflowedAngularMomentumIsGettable() Get the outflowedAngularMomentum property of the hotHalo component.

<logical> outflowedAngularMomentumIsSettable() Specify whether the outflowedAngularMomentum property of the hotHalo component is settable.

<void> outflowedAngularMomentumRate(<double> value) Cumulate to the rate of the outflowedAngularMomentum property of the HotHaloStandard component.

<void> outflowedAngularMomentumScale(<double> value) Set the scale of the outflowedAngularMomentum property of the HotHaloStandard component.

<void> outflowedAngularMomentumSet(<double> value) Set the outflowedAngularMomentum property of the hotHalo component.

<double> outflowedMass() Get the outflowedMass property of the hotHalo component.

<integer> outflowedMassCount() Compute the count of evolvable quantities in the outflowedMass property of the HotHaloStandard component.

<double> outflowedMassIsGettable() Get the outflowedMass property of the hotHalo component.

<logical> 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.

<type(abundances)> outflowingAbundances() Get the outflowingAbundances property of the hotHalo component.

<integer> outflowingAbundancesCount() Compute the count of evolvable quantities in the outflowingAbundances property of the HotHaloStandard component.

<type(abundances)> outflowingAbundancesIsGettable() Get the outflowingAbundances property of the hotHalo component.

<logical> 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 hotHalo component.

<void> outflowingAbundancesRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingAbundances property of the hotHalo component.

<logical> outflowingAbundancesRateIsAttached() Return whether the rate method of the outflowingAbundances property of the hotHalo component has been attached to a function.

<double> outflowingAngularMomentum() Get the outflowingAngularMomentum property of the hotHalo component.

<integer> outflowingAngularMomentumCount() Compute the count of evolvable quantities in the outflowingAngularMomentum property of the HotHaloStandard component.

<double> outflowingAngularMomentumIsGettable() Get the outflowingAngularMomentum property of the hotHalo component.

<logical> outflowingAngularMomentumIsSettable() Specify whether the outflowingAngularMomentum property of the hotHalo component is settable.

<void> outflowingAngularMomentumRate(<double> value) Cumulate to the rate of the outflowingAngularMomentum property of the hotHalo component.

<void> outflowingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingAngularMomentum property of the hotHalo component.

<logical> outflowingAngularMomentumRateIsAttached() Return whether the rate method of the outflowingAngularMomentum property of the hotHalo component has been attached to a function.

<double> outflowingMass() Get the outflowingMass property of the hotHalo component.

<integer> outflowingMassCount() Compute the count of evolvable quantities in the outflowingMass property of the HotHaloStandard component.

<double> outflowingMassIsGettable() Get the outflowingMass property of the hotHalo component.

<logical> outflowingMassIsSettable() Specify whether the outflowingMass property of the hotHalo component is settable.

<void> outflowingMassRate(<double> value) Cumulate to the rate of the outflowingMass property of the hotHalo component.

<void> outflowingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingMass property of the hotHalo component.

<logical> outflowingMassRateIsAttached() Return whether the rate method of the outflowingMass property of the hotHalo component has been attached to a function.

<void> output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.
15. Coding GALACTICUS

<void> outputCount(<integer> integerPropertyCount\rightarrow, <integer> doublePropertyCount\rightarrow, <double> time\rightarrow, <integer> instance\rightarrow) Compute a count of outputtable properties.

<void> outputNames(<integer> integerProperty\rightarrow, <char[*]() integerPropertyNames\rightarrow, <char[*]() integerPropertyComments\rightarrow, <double()> integerPropertyUnitsSI\rightarrow, <integer> doubleProperty\rightarrow, <char[*]() doublePropertyNames\rightarrow, <char[*]() doublePropertyComments\rightarrow, <double()> doublePropertyUnitsSI\rightarrow, <double> time\rightarrow, <integer> instance\rightarrow) Generate names of outputtable properties.

<double> potential(<double> radius\rightarrow, <componentType> [componentType]\rightarrow, <massType> [massType]\rightarrow, <logical> [haloLoaded]\rightarrow) Compute the gravitational potential.

<void> readRaw(<integer> fileHandle\rightarrow) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius\rightarrow, <componentType> [componentType]\rightarrow, <massType> [massType]\rightarrow, <logical> [haloLoaded]\rightarrow) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius\rightarrow, <componentType> [componentType]\rightarrow, <massType> [massType]\rightarrow, <logical> [haloLoaded]\rightarrow) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double()> array\rightarrow) Serialize the evolvable rates to an array.

<void> serializeScales(<double()> array\rightarrow) Serialize the evolvable scales to an array.

<void> serializeValues(<double()> array\rightarrow) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the hotHalo component class is active.

<type(abundances)> strippedAbundances() Get the strippedAbundances property of the hotHalo component.

<integer> strippedAbundancesCount() Compute the count of evolvable quantities in the strippedAbundances property of the HotHaloStandard component.

<type(abundances)> strippedAbundancesIsGettable() Get the strippedAbundances property of the hotHalo component.

<logical> strippedAbundancesIsSettable() Specify whether the strippedAbundances property of the hotHalo component is settable.

<void> strippedAbundancesRate()<type(abundances)> value) Cumulate to the rate of the strippedAbundances property of the HotHaloStandard component.

<void> strippedAbundancesScale<type(abundances)> value) Set the scale of the strippedAbundances property of the HotHaloStandard component.

<void> strippedAbundancesSet<type(abundances)> value) Set the strippedAbundances property of the hotHalo component.

<double> strippedMass() Get the strippedMass property of the hotHalo component.

<integer> strippedMassCount() Compute the count of evolvable quantities in the strippedMass property of the HotHaloStandard component.
15.5. Objects

<double> strippedMassIsGettable() Get the strippedMass property of the hotHalo component.

<logical> strippedMassIsSettable() Specify whether the strippedMass property of the hotHalo component is settable.

<void> strippedMassRate(<double> value) Cumulate to the rate of the strippedMass property of the HotHaloStandard component.

<void> strippedMassScale(<double> value) Set the scale of the strippedMass property of the HotHaloStandard component.

<void> strippedMassSet(<double> value) Set the strippedMass property of the hotHalo component.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

<double> unaccretedMass() Get the unaccretedMass property of the hotHalo component.

<integer> unaccretedMassCount() Compute the count of evolvable quantities in the unaccretedMass property of the HotHaloStandard component.

<double> unaccretedMassIsGettable() Get the unaccretedMass property of the hotHalo component.

<logical> unaccretedMassIsSettable() Specify whether the unaccretedMass property of the hotHalo component is settable.

<void> unaccretedMassRate(<double> value) Cumulate to the rate of the unaccretedMass property of the HotHaloStandard component.

<void> unaccretedMassScale(<double> value) Set the scale of the unaccretedMass property of the HotHaloStandard component.

<void> unaccretedMassSet(<double> value) Set the unaccretedMass property of the hotHalo component.

<logical> verySimpleIsActive() Return whether the verySimple implementation of the hotHalo component class is active.

nodeComponentHotHaloVerySimple

<type(abundances)> abundances() Get the abundances property of the hotHalo component.

<integer> abundancesCount() Compute the count of evolvable quantities in the abundances property of the HotHaloStandard component.

<type(abundances)> abundancesIsGettable() Get the abundances property of the hotHalo component.

<logical> abundancesIsSettable() Specify whether the abundances property of the hotHalo component is settable.
<void> abundancesRate(<type(abundances)> value) Cumulate to the rate of the abundances property of the HotHaloStandard component.

<void> abundancesScale(<type(abundances)> value) Set the scale of the abundances property of the HotHaloStandard component.

<void> abundancesSet(<type(abundances)> value) Set the abundances property of the hotHalo component.

<double> angularMomentum() Get the angularMomentum property of the hotHalo component.

<integer> angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the HotHaloStandard component.

<double> angularMomentumIsGettable() Get the angularMomentum property of the hotHalo component.

<logical> angularMomentumIsSettable() Specify whether the angularMomentum property of the hotHalo component is settable.

<void> angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the HotHaloStandard component.

<void> angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the HotHaloStandard component.

<void> angularMomentumSet(<double> value) Set the angularMomentum property of the hotHalo component.

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<type(chemicalAbundances)> chemicals() Get the chemicals property of the hotHalo component.

<integer> chemicalsCount() Compute the count of evolvable quantities in the chemicals property of the HotHaloStandard component.

<type(chemicalAbundances)> chemicalsIsGettable() Get the chemicals property of the hotHalo component.

<logical> 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.

<double> density(<double(3)> positionSpherical→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the density.
15.5. Objects

<void> deserializeRates(<double(:)> array←) Deserialze the evolvable rates from an array.

<void> deserializeScales(<double(:)> array←) Deserialze the evolvable scales from an array.

<void> deserializeValues(<double(:)> array←) Deserialze the evolvable quantities from an array.

<void> destroy() Destroy the object.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)
Compute the mass enclosed within a radius.

<double> heatSource() Get the heatSource property of the hotHalo component.

<integer> heatSourceCount() Compute the count of evolvable quantities in the heatSource property of the HotHaloStandard component.

<double> heatSourceIsGettable() Get the heatSource property of the hotHalo component.

<logical> heatSourceIsSettable() Specify whether the heatSource property of the hotHalo component is settable.

<void> heatSourceRate(<double> value) Cumulate to the rate of the heatSource property of the HotHaloStandard component.

<void> heatSourceRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the heatSource property of the hotHalo component.

<logical> heatSourceRateIsAttached() Return whether the rate method of the heatSource property of the hotHalo component has been attached to a function.

<type(treeNode)> host() Return a pointer to the host treeNode object.

<type(abundances)> hotHaloCoolingAbundances() Get the hotHaloCoolingAbundances property of the hotHalo component.

<integer> hotHaloCoolingAbundancesCount() Compute the count of evolvable quantities in the hotHaloCoolingAbundances property of the HotHaloStandard component.

<type(abundances)> hotHaloCoolingAbundancesIsGettable() Get the hotHaloCoolingAbundances property of the hotHalo component.

<logical> hotHaloCoolingAbundancesIsSettable() Specify whether the hotHaloCoolingAbundances property of the hotHalo component is settable.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAbundancesRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAbundances property of the hotHalo component.
15. Coding Galacticus

<logical> hotHaloCoolingAbundancesRateIsAttached() Return whether the rate method of the hotHaloCoolingAbundances property of the hotHalo component has been attached to a function.

<double> hotHaloCoolingAngularMomentum() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

<integer> hotHaloCoolingAngularMomentumCount() Compute the count of evolvable quantities in the hotHaloCoolingAngularMomentum property of the HotHaloStandard component.

<double> hotHaloCoolingAngularMomentumIsGettable() Get the hotHaloCoolingAngularMomentum property of the hotHalo component.

<logical> hotHaloCoolingAngularMomentumIsSettable() Specify whether the hotHaloCoolingAngularMomentum property of the hotHalo component is settable.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<logical> hotHaloCoolingAngularMomentumRateIsAttached() Return whether the rate method of the hotHaloCoolingAngularMomentum property of the hotHalo component has been attached to a function.

<double> hotHaloCoolingMass() Get the hotHaloCoolingMass property of the hotHalo component.

<integer> hotHaloCoolingMassCount() Compute the count of evolvable quantities in the hotHaloCoolingMass property of the HotHaloStandard component.

<double> hotHaloCoolingMassIsGettable() Get the hotHaloCoolingMass property of the hotHalo component.

<logical> hotHaloCoolingMassIsSettable() Specify whether the hotHaloCoolingMass property of the hotHalo component is settable.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> hotHaloCoolingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the hotHaloCoolingMass property of the hotHalo component.

<logical> hotHaloCoolingMassRateIsAttached() Return whether the rate method of the hotHaloCoolingMass property of the hotHalo component has been attached to a function.

<void> initialize() Initialize the object.

<double> mass() Get the mass property of the hotHalo component.

<integer> massCount() Compute the count of evolvable quantities in the mass property of the HotHaloStandard component.

<double> massIsGettable() Get the mass property of the hotHalo component.

<logical> massIsSettable() Specify whether the mass property of the hotHalo component is settable.
15.5. Objects

<void> massRate(<double> value) Cumulate to the rate of the mass property of the HotHaloStandard component.

<void> massScale(<double> value) Set the scale of the mass property of the HotHaloStandard component.

<void> massSet(<double> value) Set the mass property of the hotHalo component.

<double> massSink() Get the massSink property of the hotHalo component.

<double> massSinkIsGettable() Get the massSink property of the hotHalo component.

<logical> massSinkIsSettable() Specify whether the massSink property of the hotHalo component is settable.

<void> massSinkSet(<double> value) Set the massSink property of the hotHalo component.

<void> nameFromIndex(<integer> count →, <varying_string>name←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the hotHalo component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<double> outerRadius() Get the outerRadius property of the hotHalo component.

<integer> outerRadiusCount() Compute the count of evolvable quantities in the outerRadius property of the HotHaloStandard component.

<void> outerRadiusFunction(<function()> deferredFunction) Set the function to be used for the get method of the outerRadius property of the hotHalo component.

<logical> outerRadiusIsAttached() Return whether the get method of the outerRadius property of the hotHalo component has been attached to a function.

<double> outerRadiusIsGettable() Get the outerRadius property of the hotHalo component.

<logical> outerRadiusIsSettable() Specify whether the outerRadius property of the hotHalo component is settable.

<void> outerRadiusRate(<double> value) Cumulate to the rate of the outerRadius property of the HotHaloStandard component.

<void> outerRadiusScale(<double> value) Set the scale of the outerRadius property of the HotHaloStandard component.

<void> outerRadiusSet(<double> value) Set the outerRadius property of the hotHalo component.

<type(abundances)> outflowedAbundances() Get the outflowedAbundances property of the hotHalo component.

<integer> outflowedAbundancesCount() Compute the count of evolvable quantities in the outflowedAbundances property of the HotHaloStandard component.
15. Coding Galacticus

```plaintext
<type(abundances)> outflowedAbundancesIsGettable() Get the outflowedAbundances property of the hotHalo component.

<logical> outflowedAbundancesIsSettable() Specify whether the outflowedAbundances property of the hotHalo component is settable.

<void> outflowedAbundancesRate(<type(abundances)> value) Cumulate to the rate of the outflowedAbundances property of the HotHaloStandard component.

<void> outflowedAbundancesScale(<type(abundances)> value) Set the scale of the outflowedAbundances property of the HotHaloStandard component.

<void> outflowedAbundancesSet(<type(abundances)> value) Set the outflowedAbundances property of the hotHalo component.

<void> outflowedAngularMomentum() Get the outflowedAngularMomentum property of the hotHalo component.

<integer> outflowedAngularMomentumCount() Compute the count of evolvable quantities in the outflowedAngularMomentum property of the HotHaloStandard component.

<double> outflowedAngularMomentumIsGettable() Get the outflowedAngularMomentum property of the hotHalo component.

<logical> outflowedAngularMomentumIsSettable() Specify whether the outflowedAngularMomentum property of the hotHalo component is settable.

<void> outflowedAngularMomentumRate(<double> value) Cumulate to the rate of the outflowedAngularMomentum property of the HotHaloStandard component.

<void> outflowedAngularMomentumScale(<double> value) Set the scale of the outflowedAngularMomentum property of the HotHaloStandard component.

<void> outflowedAngularMomentumSet(<double> value) Set the outflowedAngularMomentum property of the hotHalo component.

<double> outflowedMass() Get the outflowedMass property of the hotHalo component.

<integer> outflowedMassCount() Compute the count of evolvable quantities in the outflowedMass property of the HotHaloStandard component.

<double> outflowedMassIsGettable() Get the outflowedMass property of the hotHalo component.

<logical> 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.

<type(abundances)> outflowingAbundances() Get the outflowingAbundances property of the hotHalo component.
```
15.5. Objects

<integer> outflowingAbundancesCount() Compute the count of evolvable quantities in the outflowingAbundances property of the HotHaloStandard component.

<type(abundances)> outflowingAbundancesIsGettable() Get the outflowingAbundances property of the hotHalo component.

<logical> 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.

<void> outflowingAbundancesRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingAbundances property of the hotHalo component.

<logical> outflowingAbundancesRateIsAttached() Return whether the rate method of the outflowingAbundances property of the hotHalo component has been attached to a function.

<double> outflowingAngularMomentum() Get the outflowingAngularMomentum property of the hotHalo component.

<integer> outflowingAngularMomentumCount() Compute the count of evolvable quantities in the outflowingAngularMomentum property of the HotHaloStandard component.

<double> outflowingAngularMomentumIsGettable() Get the outflowingAngularMomentum property of the hotHalo component.

<logical> outflowingAngularMomentumIsSettable() Specify whether the outflowingAngularMomentum property of the hotHalo component is settable.

<void> outflowingAngularMomentumRate(<double> value) Cumulate to the rate of the outflowingAngularMomentum property of the HotHaloStandard component.

<void> outflowingAngularMomentumRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingAngularMomentum property of the hotHalo component.

<logical> outflowingAngularMomentumRateIsAttached() Return whether the rate method of the outflowingAngularMomentum property of the hotHalo component has been attached to a function.

<double> outflowingMass() Get the outflowingMass property of the hotHalo component.

<integer> outflowingMassCount() Compute the count of evolvable quantities in the outflowingMass property of the HotHaloStandard component.

<double> outflowingMassIsGettable() Get the outflowingMass property of the hotHalo component.

<logical> outflowingMassIsSettable() Specify whether the outflowingMass property of the hotHalo component is settable.

<void> outflowingMassRate(<double> value) Cumulate to the rate of the outflowingMass property of the hotHalo component.

<void> outflowingMassRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the outflowingMass property of the hotHalo component.
The code snippet provided includes several functions and methods related to the Galacticus framework. Here's a breakdown of the functions and their descriptions:

1. **outflowingMassRateIsAttached()**
   - Description: Returns whether the rate method of the outflowingMass property of the hotHalo component has been attached to a function.

2. **output()**
   - Parameters: (integer) integerProperty, (integer) integerBufferCount, (integer) integerBuffer, (integer) doubleProperty, (integer) doubleBufferCount, (double) doubleBuffer, (double) time, (integer) instance
   - Description: Generates values of outputtable properties.

3. **outputCount()**
   - Parameters: (integer) integerPropertyCount, (integer) doublePropertyCount, (double) time, (integer) instance
   - Description: Computes a count of outputtable properties.

4. **outputNames()**
   - Parameters: (integer) integerProperty, (char[*]) integerPropertyNames, (char[*]) integerPropertyComments, (double[*]) integerPropertyUnitsSI, (integer) doubleProperty, (char[*]) doublePropertyNames, (char[*]) doublePropertyComments, (double[*]) doublePropertyUnitsSI, (double) time, (integer) instance
   - Description: Generates names of outputtable properties.

5. **potential()**
   - Parameters: (double) radius, (componentType) [componentType], (massType) [massType], (logical) [haloLoaded]
   - Description: Computes the gravitational potential.

6. **readRaw()**
   - Parameters: (integer) fileHandle
   - Description: Reads a binary dump of the nodeComponent from the given fileHandle.

7. **rotationCurve()**
   - Parameters: (double) radius, (componentType) [componentType], (massType) [massType], (logical) [haloLoaded]
   - Description: Computes the rotation curve.

8. **rotationCurveGradient()**
   - Parameters: (double) radius, (componentType) [componentType], (massType) [massType], (logical) [haloLoaded]
   - Description: Computes the rotation curve gradient.

9. **serializeCount()**
   - Description: Returns a count of the number of evolvable quantities to be evolved.

10. **serializeRates()**
    - Parameters: (double[*]) array
    - Description: Serializes the evolvable rates to an array.

11. **serializeScales()**
    - Parameters: (double[*]) array
    - Description: Serializes the evolvable scales to an array.

12. **serializeValues()**
    - Parameters: (double[*]) array
    - Description: Serializes the evolvable quantities to an array.

13. **standardIsActive()**
    - Description: Returns whether the standard implementation of the hotHalo component class is active.

14. **strippedAbundances()**
    - Description: Gets the strippedAbundances property of the hotHalo component.

15. **strippedAbundancesCount()**
    - Description: Computes the count of evolvable quantities in the strippedAbundances property of the HotHaloStandard component.

16. **strippedAbundancesIsGettable()**
    - Description: Gets the strippedAbundances property of the hotHalo component.

17. **strippedAbundancesIsSettable()**
    - Description: Specifies whether the strippedAbundances property of the hotHalo component is settable.

18. **strippedAbundancesRate()**
    - Parameters: (type(abundances)) value
    - Description: Cumulates to the rate of the strippedAbundances property of the HotHaloStandard component.
<void> strippedAbundancesScale(<type(abundances)> value) Set the scale of the strippedAbundances property of the HotHaloStandard component.

<void> strippedAbundancesSet(<type(abundances)> value) Set the strippedAbundances property of the hotHalo component.

<double> strippedMass() Get the strippedMass property of the hotHalo component.

<integer> strippedMassCount() Compute the count of evolvable quantities in the strippedMass property of the HotHaloStandard component.

<double> strippedMassIsGettable() Get the strippedMass property of the hotHalo component.

<logical> strippedMassIsSettable() Specify whether the strippedMass property of the hotHalo component is settable.

<void> strippedMassRate(<double> value) Cumulate to the rate of the strippedMass property of the HotHaloStandard component.

<void> strippedMassScale(<double> value) Set the scale of the strippedMass property of the HotHaloStandard component.

<void> strippedMassSet(<double> value) Set the strippedMass property of the hotHalo component.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

<double> unaccretedMass() Get the unaccretedMass property of the hotHalo component.

<integer> unaccretedMassCount() Compute the count of evolvable quantities in the unaccretedMass property of the HotHaloStandard component.

<double> unaccretedMassIsGettable() Get the unaccretedMass property of the hotHalo component.

<logical> unaccretedMassIsSettable() Specify whether the unaccretedMass property of the hotHalo component is settable.

<void> unaccretedMassRate(<double> value) Cumulate to the rate of the unaccretedMass property of the HotHaloStandard component.

<void> unaccretedMassScale(<double> value) Set the scale of the unaccretedMass property of the HotHaloStandard component.

<void> unaccretedMassSet(<double> value) Set the unaccretedMass property of the hotHalo component.

<logical> verySimpleIsActive() Return whether the verySimple implementation of the hotHalo component class is active.
15. Coding Galacticus

nodeComponentIndices

\texttt{\textbf{<integer(kind=kind\_int8)> branchTip()}} Get the \texttt{branchTip} property of the \texttt{indices} component.

\texttt{\textbf{<integer(kind=kind\_int8)> branchTipIsGettable()}} Get the \texttt{branchTip} property of the \texttt{indices} component.

\texttt{\textbf{<logical> branchTipIsSettable()}} Specify whether the \texttt{branchTip} property of the \texttt{indices} component is settable.

\texttt{\textbf{<void> branchTipSet(<integer(kind=kind\_int8)> value)}} Set the \texttt{branchTip} property of the \texttt{indices} component.

\texttt{\textbf{<void> builder(<\*type(node)>componentDefinition→)}} Build a \texttt{nodeComponent} from a supplied XML definition.

\texttt{\textbf{<double> density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)}} Compute the density.

\texttt{\textbf{<void> deserializeRates(<double(:)> array←)}} Deserialize the evolvable rates from an array.

\texttt{\textbf{<void> deserializeScales(<double(:)> array←)}} Deserialze the evolvable scales from an array.

\texttt{\textbf{<void> deserializeValues(<double(:)> array←)}} Deserialze the evolvable quantities from an array.

\texttt{\textbf{<void> destroy()}} Destroy the object.

\texttt{\textbf{<void> dump()}} Generate an ASCII dump of all properties.

\texttt{\textbf{<void> dumpRaw(<integer> fileHandle→)}} Generate a binary dump of all properties.

\texttt{\textbf{<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)}} Compute the mass enclosed within a radius.

\texttt{\textbf{<\*type(treeNode)> host()}} Return a pointer to the host \texttt{treeNode} object.

\texttt{\textbf{<void> hotHaloCoolingAbundancesRate(<type(abundances)> value)}} Cumulate to the rate of the \texttt{hotHaloCoolingAbundances} property of the \texttt{hotHalo} component.

\texttt{\textbf{<void> hotHaloCoolingAngularMomentumRate(<double> value)}} Cumulate to the rate of the \texttt{hotHaloCoolingAngularMomentum} property of the \texttt{hotHalo} component.

\texttt{\textbf{<void> hotHaloCoolingMassRate(<double> value)}} Cumulate to the rate of the \texttt{hotHaloCoolingMass} property of the \texttt{hotHalo} component.

\texttt{\textbf{<void> initialize()}} Initialize the object.

\texttt{\textbf{<void> nameFromIndex(<integer> count→, <varying_string>name←)}} Return the name of a property given its index.

\texttt{\textbf{<logical> nullIsActive()}} Return whether the null implementation of the \texttt{indices} component class is active.

\texttt{\textbf{<void> odeStepRatesInitialize()}} Initialize rates for evolvable properties.

\texttt{\textbf{<void> odeStepScalesInitialize()}} Initialize scales for evolvable properties.
15.5. Objects

```plaintext
<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)>
integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)>
doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double>
time →, <integer> instance →) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →,
<logical> [haloLoaded] →) Compute the gravitational potential.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType>
[massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType>
[massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the indices component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →,
<massType> [massType] →, <logical> [haloLoaded] →) Compute the surface density.

<type(varying_string)> type() Return the type of this object.
```

nodeComponentIndicesNull

```plaintext
<integer(kind=kind_int8)> branchTip() Get the branchTip property of the indices component.

<integer(kind=kind_int8)> branchTipIsGettable() Get the branchTip property of the indices component.

<logical> branchTipIsSettable() Specify whether the branchTip property of the indices component is settable.

<void> branchTipSet(<integer(kind=kind_int8)> value) Set the branchTip property of the indices component.

<void> builder(<*type(node)>componentDefinition →) Build a nodeComponent from a supplied XML definition.
```
<double> 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.
<void> dump() Generate an ASCII dump of all properties.
<void> dumpRaw(<integer(·)> fileHandle →) Generate a binary dump of all properties.
<double> enclosedMass(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →)
Compute the mass enclosed within a radius.
<type(treeNode)> host() Return a pointer to the host treeNode object.
<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.
<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.
<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.
<void> initialize() Initialize the object.
<void> nameFromIndex(<integer> count →, <varying_string> name ←) Return the name of a property given is index.
<logical> nullIsActive() Return whether the null implementation of the indices component class is active.
<void> odeStepRatesInitialize() Initialize rates for evolvable properties.
<void> odeStepScalesInitialize() Initialize scales for evolvable properties.
<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(·,·)> integerBuffer ↔, <integer> doubleProperty ↔, <integer> doubleBufferCount ↔, <double(·,·)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.
<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.
<void> outputNames(<integer> integerProperty ↔, <char[·]{·}> integerPropertyNames ↔, <char[·]{·}> integerPropertyNamesComments ↔, <double(·)> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char[·]{·}> doublePropertyNames ↔, <char[·]{·}> doublePropertyComments ↔, <double(·)> doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.
15.5. Objects

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the indices component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

nodeComponentIndicesStandard

<integer(kind=kind_int8)> branchTip() Get the branchTip property of the indices component.

<integer(kind=kind_int8)> branchTipIsGettable() Get the branchTip property of the indices component.

<logical> branchTipIsSettable() Specify whether the branchTip property of the indices component is settable.

<void> branchTipSet(<integer(kind=kind_int8)> value) Set the branchTip property of the indices component.

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

369
<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<<type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given its index.

<logical> nullIsActive() Return whether the null implementation of the indices component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty←, <integer> integerBufferCount↔, <integer> doubleProperty↔, <integer> doubleBufferCount↔, <double> integerBuffer↔, <double> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.
15.5. Objects

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the indices component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

nodeComponentInterOutput

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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.

<double> diskStarFormationRate() Get the diskStarFormationRate property of the interOutput component.

<integer> diskStarFormationRateCount() Compute the count of evolvable quantities in the diskStarFormationRate property of the InterOutputStandard component.

<double> diskStarFormationRateIsGettable() Get the diskStarFormationRate property of the interOutput component.

<logical> diskStarFormationRateIsSettable() Specify whether the diskStarFormationRate property of the interOutput component is settable.

<void> diskStarFormationRateRate(<double> value) Cumulate to the rate of the diskStarFormationRate property of the InterOutputStandard component.

<void> diskStarFormationRateScale(<double> value) Set the scale of the diskStarFormationRate property of the InterOutputStandard component.

<void> diskStarFormationRateSet(<double> value) Set the diskStarFormationRate property of the interOutput component.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.
15. Coding GALACTICUS

```c
<double> enclosedMass(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →)
  Compute the mass enclosed within a radius.

<double> potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →)
  Compute the gravitational potential.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →)
  Compute the rotation curve.
```

<type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(type(abundances) value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count →, <varying_string>name ←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the interOutput component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:, :)> integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:, :)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of outputtable properties.

<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.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:) array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:) array →) Serialize the evolvable scales to an array.
15.5. Objects

 serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

 spheroidStarFormationRate() Get the spheroidStarFormationRate property of the interOutput component.

 spheroidStarFormationRateCount() Compute the count of evolvable quantities in the spheroidStarFormationRate property of the InterOutputStandard component.

 spheroidStarFormationRateIsGettable() Get the spheroidStarFormationRate property of the interOutput component.

 spheroidStarFormationRateIsSettable() Specify whether the spheroidStarFormationRate property of the interOutput component is settable.

 spheroidStarFormationRateRate(<double> value) Cumulate to the rate of the spheroidStarFormationRate property of the InterOutputStandard component.

 spheroidStarFormationRateScale(<double> value) Set the scale of the spheroidStarFormationRate property of the InterOutputStandard component.

 spheroidStarFormationRateSet(<double> value) Set the spheroidStarFormationRate property of the interOutput component.

 standardIsActive() Return whether the standard implementation of the interOutput component class is active.

 surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

 type(varying_string)> type() Return the type of this object.

 nodeComponentInterOutputNull

 builder(<*:type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

 density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

 deserializeRates(<double(:)> array←) Deserialze the evolvable rates from an array.

 deserializeScales(<double(:)> array←) Deserialze the evolvable scales from an array.

 deserializeValues(<double(:)> array←) Deserialze the evolvable quantities from an array.

 destroy() Destroy the object.

 diskStarFormationRate() Get the diskStarFormationRate property of the interOutput component.

 diskStarFormationRateCount() Compute the count of evolvable quantities in the diskStarFormationRate property of the InterOutputStandard component.

 diskStarFormationRateIsGettable() Get the diskStarFormationRate property of the interOutput component.
15. Coding GALACTICUS

<logical> diskStarFormationRateIsSettable() Specify whether the diskStarFormationRate property of the interOutput component is settable.

<void> diskStarFormationRateRate(<double> value) Cumulate to the rate of the diskStarFormationRate property of the InterOutputStandard component.

<void> diskStarFormationRateScale(<double> value) Set the scale of the diskStarFormationRate property of the InterOutputStandard component.

<void> diskStarFormationRateSet(<double> value) Set the diskStarFormationRate property of the interOutput component.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the mass enclosed within a radius.

&type(treeNode)> host() Return a pointer to the host TreeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count, <varying_string>name) Return the name of a property given its index.

<logical> nullIsActive() Return whether the null implementation of the interOutput component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty, <integer> integerBufferCount, <integer(:,:) integerBuffer, <integer>doubleProperty, <integer> doubleBufferCount, <double(:,:) doubleBuffer, <double> time, <integer> instance) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount, <integer> doublePropertyCount, <double> time, <integer> instance) Compute a count of outputtable properties.

<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.
<double> potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

<void> readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double[:] > array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double[:] > array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double[:] > array →) Serialize the evolvable quantities to an array.

<double> spheroidStarFormationRate() Get the spheroidStarFormationRate property of the interOutput component.

<integer> spheroidStarFormationRateCount() Compute the count of evolvable quantities in the spheroidStarFormationRate property of the InterOutputStandard component.

<double> spheroidStarFormationRateIsGettable() Get the spheroidStarFormationRate property of the interOutput component.

<logical> spheroidStarFormationRateIsSettable() Specify whether the spheroidStarFormationRate property of the interOutput component is settable.

<void> spheroidStarFormationRateRate(<double> value) Cumulate to the rate of the spheroidStarFormationRate property of the InterOutputStandard component.

<void> spheroidStarFormationRateScale(<double> value) Set the scale of the spheroidStarFormationRate property of the InterOutputStandard component.

<void> spheroidStarFormationRateSet(<double> value) Set the spheroidStarFormationRate property of the interOutput component.

<logical> standardIsActive() Return whether the standard implementation of the interOutput component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the surface density.

<type(varying_string)> type() Return the type of this object.
nodeComponentInterOutputStandard
<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)
Compute the density.

<void> deserializeRates(<double(:)> array←) Deserializethe evolvable rates from an array.
<void> deserializeScales(<double(:)> array←) Deserializethe evolvable scales from an array.
<void> deserializeValues(<double(:)> array←) Deserializethe evolvable quantities from an array.
<void> destroy() Destroy the object.

<double> diskStarFormationRate() Get the diskStarFormationRate property of the interOutput component.
<integer> diskStarFormationRateCount() Compute the count of evolvable quantities in the diskStarFormationRate property of the InterOutputStandard component.
<double> diskStarFormationRateIsGettable() Get the diskStarFormationRate property of the interOutput component.
<logical> diskStarFormationRateIsSettable() Specify whether the diskStarFormationRate property of the interOutput component is settable.
<void> diskStarFormationRateRate(<double> value) Cumulate to the rate of the diskStarFormationRate property of the InterOutputStandard component.
<void> diskStarFormationRateScale(<double> value) Set the scale of the diskStarFormationRate property of the InterOutputStandard component.
<void> diskStarFormationRateSet(<double> value) Set the diskStarFormationRate property of the interOutput component.
<void> dump() Generate an ASCII dump of all properties.
<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.
<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)
Compute the mass enclosed within a radius.
/*type(treeNode)> host() Return a pointer to the host treeNode object.
<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.
<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.
<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.
15.5. Objects

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count, <varying_string>name←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the interOutput component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)>
integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)>
doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double>
time→, <integer> instance→) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→,
<logical> [haloLoaded]→) Compute the gravitational potential.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType>
[massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType>
[massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<double> spheroidStarFormationRate() Get the spheroidStarFormationRate property of the interOutput component.

<integer> spheroidStarFormationRateCount() Compute the count of evolvable quantities in the spheroidStarFormationRate property of the InterOutputStandard component.

<double> spheroidStarFormationRateIsGettable() Get the spheroidStarFormationRate property of the interOutput component.

<logical> spheroidStarFormationRateIsSettable() Specify whether the spheroidStarFormationRate property of the interOutput component is settable.
15. Coding GALACTICUS

```c
<void> spheroidStarFormationRateRate(<double> value) Cumulate to the rate of the spheroidStarFormationRate property of the InterOutputStandard component.

<void> spheroidStarFormationRateScale(<double> value) Set the scale of the spheroidStarFormationRate property of the InterOutputStandard component.

<void> spheroidStarFormationRateSet(<double> value) Set the spheroidStarFormationRate property of the InterOutput component.

<logical> standardIsActive() Return whether the standard implementation of the interOutput component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

nodeComponentMergingStatistics

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the density.

<void> deserializeRates(<double(:)> array←) Deserializethe evolvable rates from an array.

<void> deserializeScales(<double(:)> array←) Deserializethe evolvable scales from an array.

<void> deserializeValues(<double(:)> array←) Deserializethe evolvable quantities from an array.

<void> destroy() Destroy the object.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<double> galaxyMajorMergerTime() Get the galaxyMajorMergerTime property of the mergingStatistics component.

<double> galaxyMajorMergerTimeIsGettable() Get the galaxyMajorMergerTime property of the mergingStatistics component.

<logical> galaxyMajorMergerTimeIsSettable() Specify whether the galaxyMajorMergerTime property of the mergingStatistics component is settable.

<void> galaxyMajorMergerTimeSet(<double> value) Set the galaxyMajorMergerTime property of the mergingStatistics component.

<*type(treeNode)> host() Return a pointer to the host treeNode object.
```
15.5. Objects

```csharp
<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count, <varying_string>name) Return the name of a property given is index.

<double> nodeFormationTime() Get the nodeFormationTime property of the mergingStatistics component.

<double> nodeFormationTimeIsGettable() Get the nodeFormationTime property of the mergingStatistics component.

<logical> nodeFormationTimeIsSettable() Specify whether the nodeFormationTime property of the mergingStatistics component is settable.

<void> nodeFormationTimeSet(<double> value) Set the nodeFormationTime property of the mergingStatistics component.

<integer> nodeHierarchyLevel() Get the nodeHierarchyLevel property of the mergingStatistics component.

<integer> nodeHierarchyLevelIsGettable() Get the nodeHierarchyLevel property of the mergingStatistics component.

<logical> nodeHierarchyLevelIsSettable() Specify whether the nodeHierarchyLevel property of the mergingStatistics component is settable.

<void> nodeHierarchyLevelSet(<integer> value) Set the nodeHierarchyLevel property of the mergingStatistics component.

<double> nodeMajorMergerTime() Get the nodeMajorMergerTime property of the mergingStatistics component.

<double> nodeMajorMergerTimeIsGettable() Get the nodeMajorMergerTime property of the mergingStatistics component.

<logical> nodeMajorMergerTimeIsSettable() Specify whether the nodeMajorMergerTime property of the mergingStatistics component is settable.

<void> nodeMajorMergerTimeSet(<double> value) Set the nodeMajorMergerTime property of the mergingStatistics component.

<logical> nullIsActive() Return whether the null implementation of the mergingStatistics component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.
```
<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)>
  integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)>
  doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double>
  time →, <integer> instance →) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →,
  <logical> [haloLoaded] →) Compute the gravitational potential.

<logical> recentIsActive() Return whether the recent implementation of the mergingStatistics component class is active.

<integer(:)> recentMajorMergerCount() Get the recentMajorMergerCount property of the mergingStatistics component.

<integer(:)> recentMajorMergerCountIsGettable() Get the recentMajorMergerCount property of the mergingStatistics component.

<logical> recentMajorMergerCountIsSettable() Specify whether the recentMajorMergerCount property of the mergingStatistics component is settable.

<void> recentMajorMergerCountSet(<integer(:)> value) Set the recentMajorMergerCount property of the mergingStatistics component.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType>
  [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType>
  [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the mergingStatistics component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →,
  <massType> [massType] →, <logical> [haloLoaded] →) Compute the surface density.

<type(varying_string)> type() Return the type of this object.
nodeComponentMergingStatisticsNull

```c
<void> builder(<*type(node)> componentDefinition→) Build a nodeComponent from a supplied
XML definition.
```

```c
<double> density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType>
[massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)
Compute the density.
```

```c
<void> deserializeRates(<double(:)> array←) Deserialize the evolvable rates from an array.
```

```c
<void> deserializeScales(<double(:)> array←) Deserialize the evolvable scales from an array.
```

```c
<void> deserializeValues(<double(:)> array←) Deserializetheevolvablequantitiesfromanarray.
```

```c
<void> destroy() Destroy the object.
```

```c
<void> dump() Generate an ASCII dump of all properties.
```

```c
<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.
```

```c
<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType>
[massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)
Compute the mass enclosed within a radius.
```

```c
<double> galaxyMajorMergerTime() Get the galaxyMajorMergerTime property of the mergingStatistics
component.
```

```c
<double> galaxyMajorMergerTimeIsGettable() Get the galaxyMajorMergerTime property of the
mergingStatistics component.
```

```c
<logical> galaxyMajorMergerTimeIsSettable() Specify whether the galaxyMajorMergerTime prop-
erty of the mergingStatistics component is settable.
```

```c
<void> galaxyMajorMergerTimeSet(<double> value) Set the galaxyMajorMergerTime property of
the mergingStatistics component.
```

```c
<*type(treeNode)> host() Return a pointer to the host treeNode object.
```

```c
<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the
hotHaloCoolingAbundances property of the hotHalo component.
```

```c
<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMom-
property of the hotHalo component.
```

```c
<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass
property of the hotHalo component.
```

```c
<void> initialize() Initialize the object.
```

```c
<void> nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a prop-
erty given is index.
```

```c
<double> nodeFormationTime() Get the nodeFormationTime property of the mergingStatistics
component.
```

```c
<double> nodeFormationTimeIsGettable() Get the nodeFormationTime property of the mergingStatistics
component.
```
<logical> nodeFormationTimeIsSettable() Specify whether the nodeFormationTime property of the mergingStatistics component is settable.

<void> nodeFormationTimeSet(<double> value) Set the nodeFormationTime property of the mergingStatistics component.

<integer> nodeHierarchyLevel() Get the nodeHierarchyLevel property of the mergingStatistics component.

<integer> nodeHierarchyLevelIsGettable() Get the nodeHierarchyLevel property of the mergingStatistics component.

<logical> nodeHierarchyLevelIsSettable() Specify whether the nodeHierarchyLevel property of the mergingStatistics component is settable.

<void> nodeHierarchyLevelSet(<integer> value) Set the nodeHierarchyLevel property of the mergingStatistics component.

<double> nodeMajorMergerTime() Get the nodeMajorMergerTime property of the mergingStatistics component.

<double> nodeMajorMergerTimeIsGettable() Get the nodeMajorMergerTime property of the mergingStatistics component.

<logical> nodeMajorMergerTimeIsSettable() Specify whether the nodeMajorMergerTime property of the mergingStatistics component is settable.

<void> nodeMajorMergerTimeSet(<double> value) Set the nodeMajorMergerTime property of the mergingStatistics component.

<logical> nullIsActive() Return whether the null implementation of the mergingStatistics component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerPropertyleftrightarrow, <integer> integerBufferCountleftrightarrow, <integer(,:,:) integerBufferleftrightarrow, <integer> doublePropertyleftrightarrow, <integer> doubleBufferCountleftrightarrow, <double(,:,:) doubleBufferleftrightarrow, <double> time→, <integer> instance→) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCountleftrightarrow, <integer> doublePropertyCountleftrightarrow, <double> time→, <integer> instance→) Compute a count of outputtable properties.

<void> outputNames(<integer> integerPropertyleftrightarrow, <char[*](:) integerPropertyNamesleftrightarrow, <char[*](:) integerPropertyCommentsleftrightarrow, <double(,:) integerPropertyUnitsSIleftrightarrow, <integer> doublePropertyleftrightarrow, <char[*](:) doublePropertyNamesleftrightarrow, <char[*](:) doublePropertyCommentsleftrightarrow, <double(,:) doublePropertyUnitsSIleftrightarrow, <double> time→, <integer> instance→) Generate names of outputtable properties.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.
15.5. Objects

<logical> recentIsActive() Return whether the recent implementation of the mergingStatistics component class is active.

<integer( : )> recentMajorMergerCount() Get the recentMajorMergerCount property of the mergingStatistics component.

<integer( : )> recentMajorMergerCountIsGettable() Get the recentMajorMergerCount property of the mergingStatistics component.

<logical> recentMajorMergerCountIsSettable() Specify whether the recentMajorMergerCount property of the mergingStatistics component is settable.

<void> recentMajorMergerCountSet( <integer( : )> value ) Set the recentMajorMergerCount property of the mergingStatistics component.

<double> rotationCurve(<double( : )> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient(<double( : )> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double( : )> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double( : )> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double( : )> array →) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the mergingStatistics component class is active.

<double> surfaceDensity(<double( : )> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

nodeComponentMergingStatisticsRecent

<void> builder( <*type(node)> componentDefinition → ) Build a nodeComponent from a supplied XML definition.

<double> density(<double( : )> positionSpherical →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the density.

<void> deserializeRates(<double( : )> array ←) Deserializethe evolvable rates from an array.

<void> deserializeScales(<double( : )> array ←) Deserializethe evolvable scales from an array.

<void> deserializeValues(<double( : )> array ←) Deserializethe evolvable quantities from an array.

<void> destroy() Destroy the object.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle →) Generate a binary dump of all properties.
15. Coding GALACTICUS

\[
\text{double enclosedMass}(\text{double radius} \rightarrow, \text{componentType} [\text{componentType}] \rightarrow, \text{massType} [\text{massType}] \rightarrow, \text{weightBy} [\text{weightBy}] \rightarrow, \text{integer} [\text{weightIndex}] \rightarrow, \text{logical} [\text{haloLoaded}] \rightarrow)
\]

Compute the mass enclosed within a radius.

\[
\text{double} \text{galaxyMajorMergerTime}() \text{Get the galaxyMajorMergerTime property of the mergingStatistics component.}
\]

\[
\text{double} \text{galaxyMajorMergerTimeIsGettable}() \text{Get the galaxyMajorMergerTime property of the mergingStatistics component.}
\]

\[
\text{logical} \text{galaxyMajorMergerTimeIsSettable}() \text{Specify whether the galaxyMajorMergerTime property of the mergingStatistics component is settable.}
\]

\[
\text{void} \text{galaxyMajorMergerTimeSet}(\text{double} \text{value}) \text{Set the galaxyMajorMergerTime property of the mergingStatistics component.}
\]

\[
\text{type(treeNode)} \text{host}() \text{Return a pointer to the host treeNode object.}
\]

\[
\text{void} \text{hotHaloCoolingAbundancesRate}(\text{type(abundances)} \text{value}) \text{Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.}
\]

\[
\text{void} \text{hotHaloCoolingAngularMomentumRate}(\text{double} \text{value}) \text{Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.}
\]

\[
\text{void} \text{hotHaloCoolingMassRate}(\text{double} \text{value}) \text{Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.}
\]

\[
\text{void} \text{initialize}() \text{Initialize the object.}
\]

\[
\text{void} \text{nameFromIndex}(\text{integer} \text{count} \rightarrow, \text{varying_string} \text{name} \leftarrow) \text{Return the name of a property given is index.}
\]

\[
\text{double} \text{nodeFormationTime}() \text{Get the nodeFormationTime property of the mergingStatistics component.}
\]

\[
\text{double} \text{nodeFormationTimeIsGettable}() \text{Get the nodeFormationTime property of the mergingStatistics component.}
\]

\[
\text{logical} \text{nodeFormationTimeIsSettable}() \text{Specify whether the nodeFormationTime property of the mergingStatistics component is settable.}
\]

\[
\text{void} \text{nodeFormationTimeSet}(\text{double} \text{value}) \text{Set the nodeFormationTime property of the mergingStatistics component.}
\]

\[
\text{integer} \text{nodeHierarchyLevel}() \text{Get the nodeHierarchyLevel property of the mergingStatistics component.}
\]

\[
\text{integer} \text{nodeHierarchyLevelIsGettable}() \text{Get the nodeHierarchyLevel property of the mergingStatistics component.}
\]

\[
\text{logical} \text{nodeHierarchyLevelIsSettable}() \text{Specify whether the nodeHierarchyLevel property of the mergingStatistics component is settable.}
\]

\[
\text{void} \text{nodeHierarchyLevelSet}(\text{integer} \text{value}) \text{Set the nodeHierarchyLevel property of the mergingStatistics component.}
\]
15.5. Objects

<double> nodeMajorMergerTime() Get the nodeMajorMergerTime property of the mergingStatistics component.

<double> nodeMajorMergerTimeIsGettable() Get the nodeMajorMergerTime property of the mergingStatistics component.

<logical> nodeMajorMergerTimeIsSettable() Specify whether the nodeMajorMergerTime property of the mergingStatistics component is settable.

<void> nodeMajorMergerTimeSet(<double> value) Set the nodeMajorMergerTime property of the mergingStatistics component.

<logical> nullIsActive() Return whether the null implementation of the mergingStatistics component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:, :)> integerBuffer↔, <integer> doubleProperty↔, <integer> doubleBufferCount↔, <double(:, :)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<logical> recentIsActive() Return whether the recent implementation of the mergingStatistics component class is active.

<integer(:)> recentMajorMergerCount() Get the recentMajorMergerCount property of the mergingStatistics component.

<integer(:)> recentMajorMergerCountIsGettable() Get the recentMajorMergerCount property of the mergingStatistics component.

<logical> recentMajorMergerCountIsSettable() Specify whether the recentMajorMergerCount property of the mergingStatistics component is settable.

<void> recentMajorMergerCountSet(<integer(:)> value) Set the recentMajorMergerCount property of the mergingStatistics component.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.
15. Coding GALACTICUS

rotationCurveGradient(radius ➔, componentType ➔, massType ➔, haloLoaded ➔) Compute the rotation curve gradient.

serializeCount() Return a count of the number of evolvable quantities to be evolved.

serializeRates(array ➔) Serialize the evolvable rates to an array.

serializeScales(array ➔) Serialize the evolvable scales to an array.

serializeValues(array ➔) Serialize the evolvable quantities to an array.

standardIsActive() Return whether the standard implementation of the mergingStatistics component class is active.

surfaceDensity(positionCylindrical ➔, componentType ➔, massType ➔, haloLoaded ➔) Compute the surface density.

type() Return the type of this object.

nodeComponentMergingStatisticsStandard

builder(componentDefinition ➔) Build a nodeComponent from a supplied XML definition.

density(positionSpherical ➔, componentType ➔, massType ➔, weightBy ➔, weightIndex ➔, haloLoaded ➔) Compute the density.

deserializeRates(array ➔) Deserialize the evolvable rates from an array.

deserializeScales(array ➔) Deserialize the evolvable scales from an array.

deserializeValues(array ➔) Deserializetheevolvablequantitiesfromanarray.

destroy() Destroy the object.

dump() Generate an ASCII dump of all properties.

dumpRaw(fileHandle ➔) Generate a binary dump of all properties.

enclosedMass(radius ➔, componentType ➔, massType ➔, weightBy ➔, weightIndex ➔, haloLoaded ➔) Compute the mass enclosed within a radius.

galaxyMajorMergerTime() Get the galaxyMajorMergerTime property of the mergingStatistics component.

galaxyMajorMergerTimeIsGettable() Get the galaxyMajorMergerTime property of the mergingStatistics component.

galaxyMajorMergerTimeIsSettable() Specify whether the galaxyMajorMergerTime property of the mergingStatistics component is settable.

galaxyMajorMergerTimeSet(value) Set the galaxyMajorMergerTime property of the mergingStatistics component.

host() Return a pointer to the host treeNode object.
<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count →, <varying_string>name ←) Return the name of a property given is index.

<double> nodeFormationTime() Get the nodeFormationTime property of the mergingStatistics component.

<double> nodeFormationTimeIsGettable() Get the nodeFormationTime property of the mergingStatistics component.

<logical> nodeFormationTimeIsSettable() Specify whether the nodeFormationTime property of the mergingStatistics component is settable.

<void> nodeFormationTimeSet(<double> value) Set the nodeFormationTime property of the mergingStatistics component.

<integer> nodeHierarchyLevel() Get the nodeHierarchyLevel property of the mergingStatistics component.

<integer> nodeHierarchyLevelIsGettable() Get the nodeHierarchyLevel property of the mergingStatistics component.

<logical> nodeHierarchyLevelIsSettable() Specify whether the nodeHierarchyLevel property of the mergingStatistics component is settable.

<void> nodeHierarchyLevelSet(<integer> value) Set the nodeHierarchyLevel property of the mergingStatistics component.

<double> nodeMajorMergerTime() Get the nodeMajorMergerTime property of the mergingStatistics component.

<double> nodeMajorMergerTimeIsGettable() Get the nodeMajorMergerTime property of the mergingStatistics component.

<logical> nodeMajorMergerTimeIsSettable() Specify whether the nodeMajorMergerTime property of the mergingStatistics component is settable.

<void> nodeMajorMergerTimeSet(<double> value) Set the nodeMajorMergerTime property of the mergingStatistics component.

<logical> nullIsActive() Return whether the null implementation of the mergingStatistics component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.
15. Coding Galacticus

<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)>
integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)>
doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double>
time →, <integer> instance →) Compute a count of outputtable properties.

<void> outputNames(<integer> integerProperty ↔, <char[*](::)> integerPropertyNames ↔, <char[*](::)>
icontPropertyComments ↔, <double(:)> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔,
<double(:)> doublePropertyNames ↔, <char[*](::)> doublePropertyComments ↔, <double(:)>
doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of outputtable properties.

<double> potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →,
<logical> [haloLoaded] →) Compute the gravitational potential.

<void> readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

<logical> recentIsActive() Return whether the recent implementation of the mergingStatistics component class is active.

<integer():> recentMajorMergerCount() Get the recentMajorMergerCount property of the mergingStatistics component.

<integer():> recentMajorMergerCountIsGettable() Get the recentMajorMergerCount property of the mergingStatistics component.

<logical> recentMajorMergerCountIsSettable() Specify whether the recentMajorMergerCount property of the mergingStatistics component is settable.

<void> recentMajorMergerCountSet(<integer():> value) Set the recentMajorMergerCount property of the mergingStatistics component.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →,
<logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →,
<logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double():> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double():> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double():> array →) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the mergingStatistics component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →,
<massType> [massType] →, <logical> [haloLoaded] →) Compute the surface density.

<type(varying_string)> type() Return the type of this object.
nodeComponentPosition

**void** builder(<**type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

**double** density(<double(3)> positionSpherical→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)
Compute the density.

**void** deserializeRates(<double(:)> array←) Deserializer the evolvable rates from an array.

**void** deserializeScales(<double(:)> array←) Deserializer the evolvable scales from an array.

**void** deserializeValues(<double(:)> array←) Deserializer the evolvable quantities from an array.

**void** destroy() Destroy the object.

**void** dump() Generate an ASCII dump of all properties.

**void** dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

**double** enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)
Compute the mass enclosed within a radius.

**type(treeNode)> host() Return a pointer to the host treeNode object.

**void** hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulateto the rate of the hotHaloCoolingAbundances property of the hotHalo component.

**void** hotHaloCoolingAngularMomentumRate(<double> value) Cumulateto the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

**void** hotHaloCoolingMassRate(<double> value) Cumulateto the rate of the hotHaloCoolingMass property of the hotHalo component.

**void** initialize() Initialize the object.

**void** nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

**logical** nullIsActive() Return whether the null implementation of the position component class is active.

**void** odeStepRatesInitialize() Initialize rates for evolvable properties.

**void** odeStepScalesInitialize() Initialize scales for evolvable properties.

**void** output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

**void** outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.
15. Coding Galacticus

```c
<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 output-puttable properties.

<double(:)> position() Get the position property of the position component.

<type(history)> positionHistory() Get the positionHistory property of the position component.

<type(history)> positionHistoryIsGettable() Get the positionHistory property of the position component.

<logical> 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.

<double(:)> positionIsGettable() Get the position property of the position component.

<logical> positionIsSettable() Specify whether the position property of the position component is settable.

<void> positionSet(<double(:)> value) Set the position property of the position component.

<logical> presetIsActive() Return whether the preset implementation of the position component class is active.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

<double(:)> velocity() Get the velocity property of the position component.

<logical> velocityIsGettable() Get the velocity property of the position component.

<logical> velocityIsSettable() Specify whether the velocity property of the position component is settable.

<void> velocitySet(<double(:)> value) Set the velocity property of the position component.
```
nodeComponentPositionNull

<void> builder(<type(node)>&componentDefinition) Build a nodeComponent from a supplied XML definition.

<double> 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) Deserializetheevolvablequantitiesfromanarray.

<void> destroy() Destroy the object.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) CumulatetotherateofthehotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) CumulatetotherateofthehotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count, <varying_string>name) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the position component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty, <integer> integerBufferCount, <integer(:,:)> integerBuffer, <integer>doubleProperty, <integer> doubleBufferCount, <double(:,:)> doubleBuffer, <double> time, <integer> instance) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount, <integer> doublePropertyCount, <double> time, <integer> instance) Compute a count of outputtable properties.
15. Coding GALACTICUS

```c
<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.

<double(:)> position() Get the position property of the position component.

<type(history)> positionHistory() Get the positionHistory property of the position component.

<type(history)> positionHistoryIsGettable() Get the positionHistory property of the position component.

<logical> 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.

<double(:)> positionIsGettable() Get the position property of the position component.

<logical> positionIsSettable() Specify whether the position property of the position component is settable.

<void> positionSet(<double(:)> value) Set the position property of the position component.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<logical> presetIsActive() Return whether the preset implementation of the position component class is active.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

<double(:)> velocity() Get the velocity property of the position component.
```
15.5. Objects

- **velocityIsGettable()** Get the velocity property of the position component.
- **velocityIsSettable()** Specify whether the velocity property of the position component is settable.
- **velocitySet(double value)** Set the velocity property of the position component.

### nodeComponentPositionPreset

- **builder(**type(node)> componentDefinition→)** Build a nodeComponent from a supplied XML definition.
- **density(double 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.
- **enclosedMass(double radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→)** Compute the mass enclosed within a radius.
- **host()** Return a pointer to the host treeNode object.
- **hotHaloCoolingAbundancesRate(type(abundances)> value)** Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.
- **hotHaloCoolingAngularMomentumRate(double value)** Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.
- **hotHaloCoolingMassRate(double value)** Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.
- **initialize()** Initialize the object.
- **nameFromIndex(integer count→, varying_string name←)** Return the name of a property given is index.
- **nullIsActive()** Return whether the null implementation of the position component class is active.
- **odeStepRatesInitialize()** Initialize rates for evolvable properties.
- **odeStepScalesInitialize()** Initialize scales for evolvable properties.
15. Coding GALACTICUS

<void> output (<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer[:,:]> integerBuffer ↔, <integer> doubleProperty ↔, <integer> doubleBufferCount ↔, <double[:,:]> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of out-puttable properties.

<void> outputCount (<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →, <integer> instance →) Compute a count of out-puttable properties.

<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 out-puttable properties.

<double[:,:]> position() Get the position property of the position component.

<type(history)> positionHistory() Get the positionHistory property of the position component.

<type(history)> positionHistoryIsGettable() Get the positionHistory property of the position component.

<logical> 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.

<double[:,:]> positionIsGettable() Get the position property of the position component.

<logical> positionIsSettable() Specify whether the position property of the position component is settable.

<void> positionSet (<double[:,:]> value) Set the position property of the position component.

<double> potential (<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

<logical> presetIsActive() Return whether the preset implementation of the position component class is active.

<void> readRaw (<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve (<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient (<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates (<double[:,:]> array →) Serialize the evolvable rates to an array.

<void> serializeScales (<double[:,:]> array →) Serialize the evolvable scales to an array.
15.5. Objects

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

<double(:)> velocity() Get the velocity property of the position component.

<double(:)> velocityIsGettable() Get the velocity property of the position component.

<logical> velocityIsSettable() Specify whether the velocity property of the position component is settable.

<void> velocitySet(<double(:)> value) Set the velocity property of the position component.

nodeComponentSatellite

<double> boundMass() Get the boundMass property of the satellite component.

<integer> boundMassCount() Compute the count of evolvable quantities in the boundMass property of the SatelliteStandard component.

<type(history)> boundMassHistory() Get the boundMassHistory property of the satellite component.

<type(history)> boundMassHistoryIsGettable() Get the boundMassHistory property of the satellite component.

<logical> boundMassHistoryIsSettable() Specify whether the boundMassHistory property of the satellite component is settable.

<void> boundMassHistorySet(<type(history)> value) Set the boundMassHistory property of the satellite component.

<double> boundMassIsGettable() Get the boundMass property of the satellite component.

<logical> boundMassIsSettable() Specify whether the boundMass property of the satellite component is settable.

<void> boundMassRate(<double> value) Cumulate to the rate of the boundMass property of the SatelliteStandard component.

<void> boundMassScale(<double> value) Set the scale of the boundMass property of the SatelliteStandard component.

<void> boundMassSet(<double> value) Set the boundMass property of the satellite component.

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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.
15. Coding GALACTICUS

```c
<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.
<void> dump() Generate an ASCII dump of all properties.
<void> dumpRaw(<integer> fileHandle) Generate a binary dump of all properties.
<double> enclosedMass(<double> radius, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.
<*type(treeNode)> host() Return a pointer to the host treeNode object.
<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.
<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.
<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.
<void> initialize() Initialize the object.
<double> mergeTime() Get the mergeTime property of the satellite component.
<integer> mergeTimeCount() Compute the count of evolvable quantities in the mergeTime property of the SatelliteVerySimple component.
<double> mergeTimeIsGettable() Get the mergeTime property of the satellite component.
<logical> mergeTimeIsSettable() Specify whether the mergeTime property of the satellite component is settable.
<void> mergeTimeRate(<double> value) Cumulate to the rate of the mergeTime property of the SatelliteVerySimple component.
<void> mergeTimeScale(<double> value) Set the scale of the mergeTime property of the SatelliteVerySimple component.
<void> mergeTimeSet(<double> value) Set the mergeTime property of the satellite component.
<void> nameFromIndex(<integer> count →, <varying_string>name)→ Return the name of a property given is index.
<logical> nullIsActive() Return whether the null implementation of the satellite component class is active.
<void> odeStepRatesInitialize() Initialize rates for evolvable properties.
<void> odeStepScalesInitialize() Initialize scales for evolvable properties.
```
15.5. Objects

```c
<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)>
integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:>)
doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of out-puttable properties.

<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double>
time →, <integer> instance →) Compute a count of out-puttable properties.

<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 out-puttable properties.

<double> potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →,
<logical> [haloLoaded] →) Compute the gravitational potential.

<logical> presetIsActive() Return whether the preset implementation of the satellite component
class is active.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType>
[componentType] →, <logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →,
<massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the satellite component
class is active.

<double> surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →,
<massType> [massType] →, <logical> [haloLoaded] →) Compute the surface density.

<double> timeOfMerging() Get the timeOfMerging property of the satellite component.

<double> timeOfMergingIsGettable() Get the timeOfMerging property of the satellite component.

<logical> timeOfMergingIsSettable() Specify whether the timeOfMerging property of the satellite component
is settable.

<void> timeOfMergingSet(<double> value) Set the timeOfMerging property of the satellite component.

<type(varying_string)> type() Return the type of this object.

<logical> verySimpleIsActive() Return whether the verySimple implementation of the satellite component
class is active.
```
<type(keplerOrbit)> virialOrbit() Get the virialOrbit property of the satellite component.

<void> virialOrbitFunction(<function()> deferredFunction) Set the function to be used for the get method of the virialOrbit property of the satellite component.

<logical> virialOrbitIsAttached() Return whether the get method of the virialOrbit property of the satellite component has been attached to a function.

<type(keplerOrbit)> virialOrbitIsGettable() Get the virialOrbit property of the satellite component.

<logical> virialOrbitIsSettable() Specify whether the virialOrbit property of the satellite component is settable.

<void> virialOrbitSet(<type(keplerOrbit)> value) Set the virialOrbit property of the satellite component.

<void> virialOrbitSetFunction(<function()> deferredFunction) Set the function to be used for the set method of the virialOrbit property of the satellite component.

<logical> virialOrbitSetIsAttached() Return whether the set method of the virialOrbit property of the satellite component has been attached to a function.

nodeComponentSatelliteNull

<double> boundMass() Get the boundMass property of the satellite component.

<integer> boundMassCount() Compute the count of evolvable quantities in the boundMass property of the SatelliteStandard component.

?type(history)> boundMassHistory() Get the boundMassHistory property of the satellite component.

?type(history)> boundMassHistoryIsGettable() Get the boundMassHistory property of the satellite component.

<logical> boundMassHistoryIsSettable() Specify whether the boundMassHistory property of the satellite component is settable.

<void> boundMassHistorySet(<type(history)> value) Set the boundMassHistory property of the satellite component.

<double> boundMassIsGettable() Get the boundMass property of the satellite component.

<logical> boundMassIsSettable() Specify whether the boundMass property of the satellite component is settable.

<void> boundMassRate(<double> value) Cumulate to the rate of the boundMass property of the SatelliteStandard component.

<void> boundMassScale(<double> value) Set the scale of the boundMass property of the SatelliteStandard component.

<void> boundMassSet(<double> value) Set the boundMass property of the satellite component.
15.5. Objects

```csharp
<void> builder(<type(node)> componentDefinition) Build a nodeComponent from a supplied XML definition.

<double> 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) Deserializethe evolvable quantities from an array.

<void> destroy() Destroy the object.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →)
Compute the mass enclosed within a radius.

<typename(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulatetotherateofthe hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulatetotherateofthe hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<double> mergeTime() Get the mergeTime property of the satellite component.

<integer> mergeTimeCount() Compute the count of evolvable quantities in the mergeTime property of the SatelliteVerySimple component.

<double> mergeTimeIsGettable() Get the mergeTime property of the satellite component.

<logical> mergeTimeIsSettable() Specify whether the mergeTime property of the satellite component is settable.

<void> mergeTimeRate(<double> value) Cumulate to the rate of the mergeTime property of the SatelliteVerySimple component.

<void> mergeTimeScale(<double> value) Set the scale of the mergeTime property of the SatelliteVerySimple component.

<void> mergeTimeSet(<double> value) Set the mergeTime property of the satellite component.

<void> nameFromIndex(<integer> count →, <varying_string> name) Return the name of a property given is index.
```
nullIsActive() Return whether the null implementation of the satellite component class is active.

odeStepRatesInitialize() Initialize rates for evolvable properties.

odeStepScalesInitialize() Initialize scales for evolvable properties.

output(integer integerProperty ↔, integerBufferCount ↔, integer(:, :) integerBuffer ↔, doubleProperty ↔, doubleBufferCount ↔, double(:, :) doubleBuffer ↔, double time →, integer instance →) Generate values of outputtable properties.

outputCount(integer integerPropertyCount ↔, doublePropertyCount ↔, double time →, integer instance →) Compute a count of outputtable properties.

outputNames(integer integerProperty ↔, char[*] integerPropertyNames ↔, char[*] integerPropertyComments ↔, double(:, :) integerPropertyUnitsSI ↔, doubleProperty ↔, char[*] doublePropertyNames ↔, char[*] doublePropertyComments ↔, double(:, :) doublePropertyUnitsSI ↔, double time →, integer instance →) Generate names of outputtable properties.

potential(double radius →, componentType [componentType] →, massType [massType] →, logical [haloLoaded] →) Compute the gravitational potential.

presetIsActive() Return whether the preset implementation of the satellite component class is active.

readRaw(int fileHandle) Read a binary dump of the nodeComponent from the given fileHandle.

rotationCurve(double radius →, componentType [componentType] →, massType [massType] →, logical [haloLoaded] →) Compute the rotation curve.

rotationCurveGradient(double radius →, componentType [componentType] →, massType [massType] →, logical [haloLoaded] →) Compute the rotation curve gradient.

serializeCount() Return a count of the number of evolvable quantities to be evolved.

serializeRates(double array →) Serialize the evolvable rates to an array.

serializeScales(double array →) Serialize the evolvable scales to an array.

serializeValues(double array →) Serialize the evolvable quantities to an array.

standardIsActive() Return whether the standard implementation of the satellite component class is active.

surfaceDensity(double(3) positionCylindrical →, componentType [componentType] →, massType [massType] →, logical [haloLoaded] →) Compute the surface density.

timeOfMerging() Get the timeOfMerging property of the satellite component.

timeOfMergingIsGettable() Get the timeOfMerging property of the satellite component.

timeOfMergingIsSettable() Specify whether the timeOfMerging property of the satellite component is settable.
<void> timeOfMergingSet(<double> value) Set the timeOfMerging property of the satellite component.

<type(varying_string)> type() Return the type of this object.

<logical> verySimpleIsActive() Return whether the verySimple implementation of the satellite component class is active.

<type(keplerOrbit)> virialOrbit() Get the virialOrbit property of the satellite component.

<void> virialOrbitFunction(<function()> deferredFunction) Set the function to be used for the get method of the virialOrbit property of the satellite component.

<logical> virialOrbitIsAttached() Return whether the get method of the virialOrbit property of the satellite component has been attached to a function.

<type(keplerOrbit)> virialOrbitIsGettable() Get the virialOrbit property of the satellite component.

<logical> virialOrbitIsSettable() Specify whether the virialOrbit property of the satellite component is settable.

<void> virialOrbitSet(<type(keplerOrbit)> value) Set the virialOrbit property of the satellite component.

<void> virialOrbitSetFunction(<function()> deferredFunction) Set the function to be used for the set method of the virialOrbit property of the satellite component.

<logical> virialOrbitSetIsAttached() Return whether the set method of the virialOrbit property of the satellite component has been attached to a function.

nodeComponentSatellitePreset

<double> boundMass() Get the boundMass property of the satellite component.

<integer> boundMassCount() Compute the count of evolvable quantities in the boundMass property of the SatelliteStandard component.

<type(history)> boundMassHistory() Get the boundMassHistory property of the satellite component.

<type(history)> boundMassHistoryIsGettable() Get the boundMassHistory property of the satellite component.

<logical> boundMassHistoryIsSettable() Specify whether the boundMassHistory property of the satellite component is settable.

<void> boundMassHistorySet(<type(history)> value) Set the boundMassHistory property of the satellite component.

<double> boundMassIsGettable() Get the boundMass property of the satellite component.

<logical> boundMassIsSettable() Specify whether the boundMass property of the satellite component is settable.
<void> boundMassRate(<double> value) Cumulate to the rate of the boundMass property of the SatelliteStandard component.

<void> boundMassScale(<double> value) Set the scale of the boundMass property of the SatelliteStandard component.

<void> boundMassSet(<double> value) Set the boundMass property of the satellite component.

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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←) Deserializethe evolvable quantities from an array.

<void> destroy() Destroy the object.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the mass enclosed within a radius.

<*type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundancesRate property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<double> mergeTime() Get the mergeTime property of the satellite component.

<integer> mergeTimeCount() Compute the count of evolvable quantities in the mergeTime property of the SatelliteVerySimple component.

<double> mergeTimeIsGettable() Get the mergeTime property of the satellite component.

<logical> mergeTimeIsSettable() Specify whether the mergeTime property of the satellite component is settable.

<void> mergeTimeRate(<double> value) Cumulate to the rate of the mergeTime property of the SatelliteVerySimple component.
15.5. Objects

<void> mergeTimeScale(<double> value) Set the scale of the mergeTime property of the SatelliteVerySimple component.

<void> mergeTimeSet(<double> value) Set the mergeTime property of the satellite component.

<void> nameFromIndex(<integer> count →, <varying_string>name←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the satellite component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty←, <integer> integerBufferCount←, <integer(:,:)> integerBuffer←, <integer>doubleProperty←, <integer> doubleBufferCount←, <double(:,:)> doubleBuffer←, <double> time→, <integer> instance→) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount←, <integer> doublePropertyCount←, <double> time→, <integer> instance→) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<logical> presetIsActive() Return whether the preset implementation of the satellite component class is active.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:>) array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:>) array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:>) array→) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the satellite component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.
15. Coding Galacticus

<double> timeOfMerging() Get the timeOfMerging property of the satellite component.

<double> timeOfMergingIsGettable() Get the timeOfMerging property of the satellite component.

<logical> timeOfMergingIsSettable() Specify whether the timeOfMerging property of the satellite component is settable.

<void> timeOfMergingSet(<double> value) Set the timeOfMerging property of the satellite component.

<type(varying_string)> type() Return the type of this object.

<logical> verySimpleIsActive() Return whether the verySimple implementation of the satellite component class is active.

<type(keplerOrbit)> virialOrbit() Get the virialOrbit property of the satellite component.

<void> virialOrbitFunction(<function()> deferredFunction) Set the function to be used for the get method of the virialOrbit property of the satellite component.

<logical> virialOrbitIsAttached() Return whether the get method of the virialOrbit property of the satellite component has been attached to a function.

<type(keplerOrbit)> virialOrbitIsGettable() Get the virialOrbit property of the satellite component.

<logical> virialOrbitIsSettable() Specify whether the virialOrbit property of the satellite component is settable.

<void> virialOrbitSet(<type(keplerOrbit)> value) Set the virialOrbit property of the satellite component.

<void> virialOrbitSetFunction(<function()> deferredFunction) Set the function to be used for the set method of the virialOrbit property of the satellite component.

<logical> virialOrbitSetIsAttached() Return whether the set method of the virialOrbit property of the satellite component has been attached to a function.

nodeComponentSatelliteStandard

<double> boundMass() Get the boundMass property of the satellite component.

<integer> boundMassCount() Compute the count of evolvable quantities in the boundMass property of the SatelliteStandard component.

<type(history)> boundMassHistory() Get the boundMassHistory property of the satellite component.

<type(history)> boundMassHistoryIsGettable() Get the boundMassHistory property of the satellite component.

<logical> boundMassHistoryIsSettable() Specify whether the boundMassHistory property of the satellite component is settable.
15.5. Objects

```c
<void> boundMassHistorySet(<type(history)> value) Set the boundMassHistory property of the satellite component.

<double> boundMassIsGettable() Get the boundMass property of the satellite component.

<logical> boundMassIsSettable() Specify whether the boundMass property of the satellite component is settable.

<void> boundMassRate(<double> value) Cumulate to the rate of the boundMass property of the SatelliteStandard component.

<void> boundMassScale(<double> value) Set the scale of the boundMass property of the SatelliteStandard component.

<void> boundMassSet(<double> value) Set the boundMass property of the satellite component.

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType>[componentType]→, <massType>[massType]→, <weightBy>[weightBy]→, <integer>[weightIndex]→, <logical>[haloLoaded]→) Compute the mass enclosed within a radius.

<*type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<double> mergeTime() Get the mergeTime property of the satellite component.

<integer> mergeTimeCount() Compute the count of evolvable quantities in the mergeTime property of the SatelliteVerySimple component.
```
15. Coding Galacticus

<double> mergeTimeIsGettable() Get the mergeTime property of the satellite component.

<logical> mergeTimeIsSettable() Specify whether the mergeTime property of the satellite component is settable.

<void> mergeTimeRate(<double> value) Cumulate to the rate of the mergeTime property of the SatelliteVerySimple component.

<void> mergeTimeScale(<double> value) Set the scale of the mergeTime property of the SatelliteVerySimple component.

<void> mergeTimeSet(<double> value) Set the mergeTime property of the satellite component.

<void> nameFromIndex(<integer> count →, <varying_string>name←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the satellite component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)>
integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)>
doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double>
time→, <integer> instance→) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→,
<logical> [haloLoaded]→) Compute the gravitational potential.

<logical> presetIsActive() Return whether the preset implementation of the satellite component class is active.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→,
<logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→,
<logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.
15.5. Objects

<void> serializeScales(<double(,:) array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(,:) array→) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the satellite component class is active.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<double> timeOfMerging() Get the timeOfMerging property of the satellite component.

<double> timeOfMergingIsGettable() Get the timeOfMerging property of the satellite component.

<logical> timeOfMergingIsSettable() Specify whether the timeOfMerging property of the satellite component is settable.

<void> timeOfMergingSet(<double> value) Set the timeOfMerging property of the satellite component.

<type(varying_string)> type() Return the type of this object.

<logical> verySimpleIsActive() Return whether the verySimple implementation of the satellite component class is active.

<type(keplerOrbit)> virialOrbit() Get the virialOrbit property of the satellite component.

<void> virialOrbitFunction(<function()> deferredFunction) Set the function to be used for the get method of the virialOrbit property of the satellite component.

<logical> virialOrbitIsAttached() Return whether the get method of the virialOrbit property of the satellite component has been attached to a function.

<type(keplerOrbit)> virialOrbitIsGettable() Get the virialOrbit property of the satellite component.

<logical> virialOrbitIsSettable() Specify whether the virialOrbit property of the satellite component is settable.

<void> virialOrbitSet(<type(keplerOrbit)> value) Set the virialOrbit property of the satellite component.

<void> virialOrbitSetFunction(<function()> deferredFunction) Set the function to be used for the set method of the virialOrbit property of the satellite component.

<logical> virialOrbitSetIsAttached() Return whether the set method of the virialOrbit property of the satellite component has been attached to a function.

<void> virialOrbitSetValue(<type(keplerOrbit)> value) Set the virialOrbit property of the satellite component.

<type(keplerOrbit)> virialOrbitValue() Get the virialOrbit property of the satellite component.
nodeComponentSatelliteVerySimple

<double> boundMass() Get the boundMass property of the satellite component.

<integer> boundMassCount() Compute the count of evolvable quantities in the boundMass property of the SatelliteStandard component.

<type(history)> boundMassHistory() Get the boundMassHistory property of the satellite component.

<type(history)> boundMassHistoryIsGettable() Get the boundMassHistory property of the satellite component.

<type(history)> boundMassHistoryIsSettable() Specify whether the boundMassHistory property of the satellite component is settable.

<void> boundMassHistorySet(<type(history)> value) Set the boundMassHistory property of the satellite component.

<double> boundMassIsGettable() Get the boundMass property of the satellite component.

<logical> boundMassIsSettable() Specify whether the boundMass property of the satellite component is settable.

<void> boundMassRate(<double> value) Cumulate to the rate of the boundMass property of the SatelliteStandard component.

<void> boundMassScale(<double> value) Set the scale of the boundMass property of the SatelliteStandard component.

<void> boundMassSet(<double> value) Set the boundMass property of the satellite component.

<void> builder(<*type(node)> componentDefinition →) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle →) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <weightBy> [weightBy] →, <integer> [weightIndex] →, <logical> [haloLoaded] →) Compute the mass enclosed within a radius.

<*type(treeNode)> host() Return a pointer to the host treeNode object.
<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<double> mergeTime() Get the mergeTime property of the satellite component.

<integer> mergeTimeCount() Compute the count of evolvable quantities in the mergeTime property of the SatelliteVerySimple component.

<double> mergeTimeIsGettable() Get the mergeTime property of the satellite component.

<logical> mergeTimeIsSettable() Specify whether the mergeTime property of the satellite component is settable.

<void> mergeTimeRate(<double> value) Cumulate to the rate of the mergeTime property of the SatelliteVerySimple component.

<void> mergeTimeScale(<double> value) Set the scale of the mergeTime property of the SatelliteVerySimple component.

<void> mergeTimeSet(<double> value) Set the mergeTime property of the satellite component.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.
\texttt{\textless logical\textgreater\ presetIsActive()} Return whether the preset implementation of the satellite component class is active.

\texttt{\textless void\textgreater\ readRaw(<integer> fileHandle \rightarrow)} Read a binary dump of the \texttt{nodeComponent} from the given fileHandle.

\texttt{\textless double\textgreater\ rotationCurve(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow)} Compute the rotation curve.

\texttt{\textless double\textgreater\ rotationCurveGradient(<double> radius \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow)} Compute the rotation curve gradient.

\texttt{\textless integer\textgreater\ serializeCount()} Return a count of the number of evolvable quantities to be evolved.

\texttt{\textless void\textgreater\ serializeRates(<double(:)> array \rightarrow)} Serialize the evolvable rates to an array.

\texttt{\textless void\textgreater\ serializeScales(<double(:)> array \rightarrow)} Serialize the evolvable scales to an array.

\texttt{\textless void\textgreater\ serializeValues(<double(:)> array \rightarrow)} Serialize the evolvable quantities to an array.

\texttt{\textless logical\textgreater\ standardIsActive()} Return whether the standard implementation of the satellite component class is active.

\texttt{\textless double\textgreater\ surfaceDensity(<double(3)> positionCylindrical \rightarrow, <componentType> [componentType] \rightarrow, <massType> [massType] \rightarrow, <logical> [haloLoaded] \rightarrow)} Compute the surface density.

\texttt{\textless double\textgreater\ timeOfMerging()} Get the \texttt{timeOfMerging} property of the \texttt{satellite} component.

\texttt{\textless double\textgreater\ timeOfMergingIsGettable()} Get the \texttt{timeOfMerging} property of the \texttt{satellite} component.

\texttt{\textless logical\textgreater\ timeOfMergingIsSettable()} Specify whether the \texttt{timeOfMerging} property of the \texttt{satellite} component is settable.

\texttt{\textless void\textgreater\ timeOfMergingSet(<double> value)} Set the \texttt{timeOfMerging} property of the \texttt{satellite} component.

\texttt{\textless type(varying\_string)\textgreater\ type()} Return the type of this object.

\texttt{\textless logical\textgreater\ verySimpleIsActive()} Return whether the \texttt{verySimple} implementation of the satellite component class is active.

\texttt{\textless type(kepler\_orbit)\textgreater\ virial\_orbit()} Get the \texttt{virial\_orbit} property of the \texttt{satellite} component.

\texttt{\textless void\textgreater\ virial\_orbit\_function(<function()> deferredFunction)} Set the function to be used for the \texttt{get} method of the \texttt{virial\_orbit} property of the \texttt{satellite} component.

\texttt{\textless logical\textgreater\ virial\_orbit\_is\_attached()} Return whether the \texttt{get} method of the \texttt{virial\_orbit} property of the satellite component has been attached to a function.

\texttt{\textless type(kepler\_orbit)\textgreater\ virial\_orbit\_is\_gettable()} Get the \texttt{virial\_orbit} property of the \texttt{satellite} component.

\texttt{\textless logical\textgreater\ virial\_orbit\_is\_settable()} Specify whether the \texttt{virial\_orbit} property of the \texttt{satellite} component is settable.
<void> virialOrbitSet(<type(keplerOrbit)> value) Set the virialOrbit property of the satellite component.

<void> virialOrbitSetFunction(<function()> deferredFunction) Set the function to be used for the set method of the virialOrbit property of the satellite component.

<logical> virialOrbitSetIsAttached() Return whether the set method of the virialOrbit property of the satellite component has been attached to a function.

nodeComponentSpheroid

<type(abundances)> abundancesGas() Get the abundancesGas property of the spheroid component.

<integer> abundancesGasCount() Compute the count of evolvable quantities in the abundancesGas property of the SpheroidStandard component.

<type(abundances)> abundancesGasIsGettable() Get the abundancesGas property of the spheroid component.

<logical> abundancesGasIsSettable() Specify whether the abundancesGas property of the spheroid component is settable.

<void> abundancesGasRate(<type(abundances)> value) Cumulate to the rate of the abundancesGas property of the SpheroidStandard component.

<void> abundancesGasScale(<type(abundances)> value) Set the scale of the abundancesGas property of the SpheroidStandard component.

<void> abundancesGasSet(<type(abundances)> value) Set the abundancesGas property of the spheroid component.

<type(abundances)> abundancesStellar() Get the abundancesStellar property of the spheroid component.

<integer> abundancesStellarCount() Compute the count of evolvable quantities in the abundancesStellar property of the SpheroidStandard component.

<type(abundances)> abundancesStellarIsGettable() Get the abundancesStellar property of the spheroid component.

<logical> abundancesStellarIsSettable() Specify whether the abundancesStellar property of the spheroid component is settable.

<void> abundancesStellarRate(<type(abundances)> value) Cumulate to the rate of the abundancesStellar property of the SpheroidStandard component.

<void> abundancesStellarScale(<type(abundances)> value) Set the scale of the abundancesStellar property of the SpheroidStandard component.

<void> abundancesStellarSet(<type(abundances)> value) Set the abundancesStellar property of the spheroid component.

<double> angularMomentum() Get the angularMomentum property of the spheroid component.

<integer> angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the SpheroidStandard component.
15. Coding GALACTICUS

<double> angularMomentumIsGettable() Get the angularMomentum property of the spheroid component.

<logical> angularMomentumIsSettable() Specify whether the angularMomentum property of the spheroid component is settable.

<void> angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the SpheroidStandard component.

<void> angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the SpheroidStandard component.

<void> angularMomentumSet(<double> value) Set the angularMomentum property of the spheroid component.

<void> builder(<type(node)>componentDefinition →) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle →) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius →), <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<double> energyGasInput() Get the energyGasInput property of the spheroid component.

<integer> energyGasInputCount() Compute the count of evolvable quantities in the energyGasInput property of the SpheroidStandard component.

<double> energyGasInputIsGettable() Get the energyGasInput property of the spheroid component.

<logical> energyGasInputIsSettable() Specify whether the energyGasInput property of the spheroid component is settable.

<void> energyGasInputRate(<double> value) Cumulate to the rate of the energyGasInput property of the SpheroidStandard component.

<void> energyGasInputRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the energyGasInput property of the spheroid component.

<logical> energyGasInputRateIsAttached() Return whether the rate method of the energyGasInput property of the spheroid component has been attached to a function.
15.5. Objects

<double> halfMassRadius() Get the halfMassRadius property of the spheroid component.

<double> halfMassRadiusIsGettable() Get the halfMassRadius property of the spheroid component.

<logical> halfMassRadiusIsSettable() Specify whether the halfMassRadius property of the spheroid component is settable.

*>*type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<logical> isInitialized() Get the isInitialized property of the spheroid component.

<logical> isInitializedIsGettable() Get the isInitialized property of the spheroid component.

<logical> isInitializedIsSettable() Specify whether the isInitialized property of the spheroid component is settable.

<void> isInitializedSet(<logical> value) Set the isInitialized property of the spheroid component.

<double(:)> luminositiesStellar() Get the luminositiesStellar property of the spheroid component.

<integer> luminositiesStellarCount() Compute the count of evolvable quantities in the luminositiesStellar property of the SpheroidStandard component.

<double(:)> luminositiesStellarIsGettable() Get the luminositiesStellar property of the spheroid component.

<logical> luminositiesStellarIsSettable() Specify whether the luminositiesStellar property of the spheroid component is settable.

<void> luminositiesStellarRate(<double(:)> value) Cumulate to the rate of the luminositiesStellar property of the SpheroidStandard component.

<void> luminositiesStellarScale(<double(:)> value) Set the scale of the luminositiesStellar property of the SpheroidStandard component.

<void> luminositiesStellarSet(<double(:)> value) Set the luminositiesStellar property of the spheroid component.

<double> massGas() Get the massGas property of the spheroid component.

<integer> massGasCount() Compute the count of evolvable quantities in the massGas property of the SpheroidStandard component.
<double> massGasIsGettable() Get the massGas property of the spheroid component.

<logical> massGasIsSettable() Specify whether the massGas property of the spheroid component is settable.

<void> massGasRate(<double> value) Cumulate to the rate of the massGas property of the SpheroidStandard component.

<void> massGasScale(<double> value) Set the scale of the massGas property of the SpheroidStandard component.

<void> massGasSet(<double> value) Set the massGas property of the spheroid component.

<double> massGasSink() Get the massGasSink property of the spheroid component.

<integer> massGasSinkCount() Compute the count of evolvable quantities in the massGasSink property of the SpheroidStandard component.

<double> massGasSinkIsGettable() Get the massGasSink property of the spheroid component.

<logical> massGasSinkIsSettable() Specify whether the massGasSink property of the spheroid component is settable.

<void> massGasSinkRate(<double> value) Cumulate to the rate of the massGasSink property of the SpheroidStandard component.

<void> massGasSinkRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the massGasSink property of the spheroid component.

<logical> massGasSinkRateIsAttached() Return whether the rate method of the massGasSink property of the spheroid component has been attached to a function.

<double> massStellar() Get the massStellar property of the spheroid component.

<integer> massStellarCount() Compute the count of evolvable quantities in the massStellar property of the SpheroidStandard component.

<double> massStellarIsGettable() Get the massStellar property of the spheroid component.

<logical> massStellarIsSettable() Specify whether the massStellar property of the spheroid component is settable.

<void> massStellarRate(<double> value) Cumulate to the rate of the massStellar property of the SpheroidStandard component.

<void> massStellarScale(<double> value) Set the scale of the massStellar property of the SpheroidStandard component.

<void> massStellarSet(<double> value) Set the massStellar property of the spheroid component.

<void> nameFromIndex(<integer> count →, <varying_string>name ←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the spheroid component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.
<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)>
integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)>
doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double>
time→, <integer> instance→) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→,
<logical> [haloLoaded]→) Compute the gravitational potential.

<double> radius() Get the radius property of the spheroid component.

<double> radiusIsGettable() Get the radius property of the spheroid component.

<logical> radiusIsSettable() Specify whether the radius property of the spheroid component is settable.

<void> radiusSet(<double> value) Set the radius property of the spheroid component.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> 
[massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> 
[massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the spheroid component class is active.

<type(history)> starFormationHistory() Get the starFormationHistory property of the spheroid component.

<integer> starFormationHistoryCount() Compute the count of evolvable quantities in the starFormationHistory property of the SpheroidStandard component.

<type(history)> starFormationHistoryIsGettable() Get the starFormationHistory property of the spheroid component.

<logical> starFormationHistoryIsSettable() Specify whether the starFormationHistory property of the spheroid component is settable.
15. Coding GALACTICUS

```c
<void> starFormationHistoryRate(<type(history)> value) Cumulate to the rate of the starFormationHistory property of the SpheroidStandard component.

<void> starFormationHistoryScale(<type(history)> value) Set the scale of the starFormationHistory property of the SpheroidStandard component.

<void> starFormationHistorySet(<type(history)> value) Set the starFormationHistory property of the spheroid component.

<double> starFormationRate() Get the starFormationRate property of the spheroid component.

<void> starFormationRateFunction(<function()> deferredFunction) Set the function to be used for the get method of the starFormationRate property of the spheroid component.

<logical> starFormationRateIsAttached() Return whether the get method of the starFormationRate property of the spheroid component has been attached to a function.

<double> starFormationRateIsGettable() Get the starFormationRate property of the spheroid component.

<logical> starFormationRateIsSettable() Specify whether the starFormationRate property of the spheroid component is settable.

<type(history)> stellarPropertiesHistory() Get the stellarPropertiesHistory property of the spheroid component.

<integer> stellarPropertiesHistoryCount() Compute the count of evolvable quantities in the stellarPropertiesHistory property of the SpheroidStandard component.

<type(history)> stellarPropertiesHistoryIsGettable() Get the stellarPropertiesHistory property of the spheroid component.

<logical> stellarPropertiesHistoryIsSettable() Specify whether the stellarPropertiesHistory property of the spheroid component is settable.

<void> stellarPropertiesHistoryRate(<type(history)> value) Cumulate to the rate of the stellarPropertiesHistory property of the SpheroidStandard component.

<void> stellarPropertiesHistoryScale(<type(history)> value) Set the scale of the stellarPropertiesHistory property of the SpheroidStandard component.

<void> stellarPropertiesHistorySet(<type(history)> value) Set the stellarPropertiesHistory property of the spheroid component.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

<double> velocity() Get the velocity property of the spheroid component.

<double> velocityIsGettable() Get the velocity property of the spheroid component.

<logical> velocityIsSettable() Specify whether the velocity property of the spheroid component is settable.

<void> velocitySet(<double> value) Set the velocity property of the spheroid component.
```
nodeComponentSpheroidNull

<type(abundances)> abundancesGas() Get the abundancesGas property of the spheroid component.

<integer> abundancesGasCount() Compute the count of evolvable quantities in the abundancesGas property of the SpheroidStandard component.

<type(abundances)> abundancesGasIsGettable() Get the abundancesGas property of the spheroid component.

<logical> abundancesGasIsSettable() Specify whether the abundancesGas property of the spheroid component is settable.

<void> abundancesGasRate(<type(abundances)> value) Cumulate to the rate of the abundancesGas property of the SpheroidStandard component.

<void> abundancesGasScale(<type(abundances)> value) Set the scale of the abundancesGas property of the SpheroidStandard component.

<void> abundancesGasSet(<type(abundances)> value) Set the abundancesGas property of the spheroid component.

<type(abundances)> abundancesStellar() Get the abundancesStellar property of the spheroid component.

<integer> abundancesStellarCount() Compute the count of evolvable quantities in the abundancesStellar property of the SpheroidStandard component.

<type(abundances)> abundancesStellarIsGettable() Get the abundancesStellar property of the spheroid component.

<logical> abundancesStellarIsSettable() Specify whether the abundancesStellar property of the spheroid component is settable.

<void> abundancesStellarRate(<type(abundances)> value) Cumulate to the rate of the abundancesStellar property of the SpheroidStandard component.

<void> abundancesStellarScale(<type(abundances)> value) Set the scale of the abundancesStellar property of the SpheroidStandard component.

<void> abundancesStellarSet(<type(abundances)> value) Set the abundancesStellar property of the spheroid component.

<double> angularMomentum() Get the angularMomentum property of the spheroid component.

<integer> angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the SpheroidStandard component.

<double> angularMomentumIsGettable() Get the angularMomentum property of the spheroid component.

<logical> angularMomentumIsSettable() Specify whether the angularMomentum property of the spheroid component is settable.

<void> angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the SpheroidStandard component.
15. Coding Galacticus

```c
<void> angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the SpheroidStandard component.
<void> angularMomentumSet(<double> value) Set the angularMomentum property of the spheroid component.
<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.
<double> 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.
<void> dump() Generate an ASCII dump of all properties.
<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.
<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.
<double> energyGasInput() Get the energyGasInput property of the spheroid component.
<integer> energyGasInputCount() Compute the count of evolvable quantities in the energyGasInput property of the SpheroidStandard component.
<double> energyGasInputIsGettable() Get the energyGasInput property of the spheroid component.
<logical> energyGasInputIsSettable() Specify whether the energyGasInput property of the spheroid component is settable.
<void> energyGasInputRate(<double> value) Cumulate to the rate of the energyGasInput property of the SpheroidStandard component.
<void> energyGasInputRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the energyGasInput property of the spheroid component.
<logical> energyGasInputRateIsAttached() Return whether the rate method of the energyGasInput property of the spheroid component has been attached to a function.
<double> halfMassRadius() Get the halfMassRadius property of the spheroid component.
<double> halfMassRadiusIsGettable() Get the halfMassRadius property of the spheroid component.
<logical> halfMassRadiusIsSettable() Specify whether the halfMassRadius property of the spheroid component is settable.
```
15.5. Objects

```c
// type(treeNode) host() Return a pointer to the host treeNode object.

// void hotHaloCoolingAbundancesRate(type(abundances) value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

// void hotHaloCoolingAngularMomentumRate(double value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

// void hotHaloCoolingMassRate(double value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

// void initialize() Initialize the object.

// logical isInitialized() Get the isInitialized property of the spheroid component.

// logical isInitializedIsGettable() Get the isInitialized property of the spheroid component.

// logical isInitializedIsSettable() Specify whether the isInitialized property of the spheroid component is settable.

// void isInitializedSet(logical value) Set the isInitialized property of the spheroid component.

// double(:) luminositiesStellar() Get the luminositiesStellar property of the spheroid component.

// integer luminositiesStellarCount() Compute the count of evolvable quantities in the luminositiesStellar property of the SpheroidStandard component.

// double(:) luminositiesStellarIsGettable() Get the luminositiesStellar property of the spheroid component.

// logical luminositiesStellarIsSettable() Specify whether the luminositiesStellar property of the spheroid component is settable.

// void luminositiesStellarRate(double value) Cumulate to the rate of the luminositiesStellar property of the SpheroidStandard component.

// void luminositiesStellarScale(double value) Set the scale of the luminositiesStellar property of the SpheroidStandard component.

// void luminositiesStellarSet(double value) Set the luminositiesStellar property of the spheroid component.

// double massGas() Get the massGas property of the spheroid component.

// integer massGasCount() Compute the count of evolvable quantities in the massGas property of the SpheroidStandard component.

// double massGasIsGettable() Get the massGas property of the spheroid component.

// logical massGasIsSettable() Specify whether the massGas property of the spheroid component is settable.

// void massGasRate(double value) Cumulate to the rate of the massGas property of the SpheroidStandard component.
```
15. Coding GALACTICUS

```
<void> massGasScale(<double> value) Set the scale of the massGas property of the SpheroidStandard component.

<void> massGasSet(<double> value) Set the massGas property of the spheroid component.

<double> massGasSink() Get the massGasSink property of the spheroid component.

<integer> massGasSinkCount() Compute the count of evolvable quantities in the massGasSink property of the SpheroidStandard component.

<double> massGasSinkIsGettable() Get the massGasSink property of the spheroid component.

<logical> massGasSinkIsSettable() Specify whether the massGasSink property of the spheroid component is settable.

<void> massGasSinkRate(<double> value) Cumulate to the rate of the massGasSink property of the SpheroidStandard component.

<void> massGasSinkRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the massGasSink property of the spheroid component.

<logical> massGasSinkRateIsAttached() Return whether the rate method of the massGasSink property of the spheroid component has been attached to a function.

<double> massStellar() Get the massStellar property of the spheroid component.

<integer> massStellarCount() Compute the count of evolvable quantities in the massStellar property of the SpheroidStandard component.

<double> massStellarIsGettable() Get the massStellar property of the spheroid component.

<logical> massStellarIsSettable() Specify whether the massStellar property of the spheroid component is settable.

<void> massStellarRate(<double> value) Cumulate to the rate of the massStellar property of the SpheroidStandard component.

<void> massStellarScale(<double> value) Set the scale of the massStellar property of the SpheroidStandard component.

<void> massStellarSet(<double> value) Set the massStellar property of the spheroid component.

<double> nameFromIndex(<integer> count →, <varying_string>name ←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the spheroid component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer> doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time →, <integer> instance →) Generate values of outputtable properties.
```
15.5. Objects

<void> outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<double> radius() Get the radius property of the spheroid component.

<double> radiusIsGettable() Get the radius property of the spheroid component.

<logical> radiusIsSettable() Specify whether the radius property of the spheroid component is settable.

<void> radiusSet(<double> value) Set the radius property of the spheroid component.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the spheroid component class is active.

<type(history)> starFormationHistory() Get the starFormationHistory property of the spheroid component.

<integer> starFormationHistoryCount() Compute the count of evolvable quantities in the starFormationHistory property of the SpheroidStandard component.

<type(history)> starFormationHistoryIsGettable() Get the starFormationHistory property of the spheroid component.

<logical> starFormationHistoryIsSettable() Specify whether the starFormationHistory property of the spheroid component is settable.

<void> starFormationHistoryRate(<type(history)> value) Cumulate to the rate of the starFormationHistory property of the SpheroidStandard component.
15. Coding GALACTICUS

```csharp
<void> starFormationHistoryScale(<type(history)> value) Set the scale of the starFormationHistory property of the SpheroidStandard component.

<void> starFormationHistorySet(<type(history)> value) Set the starFormationHistory property of the spheroid component.

<double> starFormationRate() Get the starFormationRate property of the spheroid component.

<void> starFormationRateFunction(<function()> deferredFunction) Set the function to be used for the get method of the starFormationRate property of the spheroid component.

<logical> starFormationRateIsAttached() Return whether the get method of the starFormationRate property of the spheroid component has been attached to a function.

<double> starFormationRateIsGettable() Get the starFormationRate property of the spheroid component.

<logical> starFormationRateIsSettable() Specify whether the starFormationRate property of the spheroid component is settable.

<type(history)> stellarPropertiesHistory() Get the stellarPropertiesHistory property of the spheroid component.

<integer> stellarPropertiesHistoryCount() Compute the count of evolvable quantities in the stellarPropertiesHistory property of the SpheroidStandard component.

<type(history)> stellarPropertiesHistoryIsGettable() Get the stellarPropertiesHistory property of the spheroid component.

<logical> stellarPropertiesHistoryIsSettable() Specify whether the stellarPropertiesHistory property of the spheroid component is settable.

<void> stellarPropertiesHistoryRate(<type(history)> value) Cumulate to the rate of the stellarPropertiesHistory property of the SpheroidStandard component.

<void> stellarPropertiesHistoryScale(<type(history)> value) Set the scale of the stellarPropertiesHistory property of the SpheroidStandard component.

<void> stellarPropertiesHistorySet(<type(history)> value) Set the stellarPropertiesHistory property of the spheroid component.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

<double> velocity() Get the velocity property of the spheroid component.

<double> velocityIsGettable() Get the velocity property of the spheroid component.

<logical> velocityIsSettable() Specify whether the velocity property of the spheroid component is settable.

<void> velocitySet(<double> value) Set the velocity property of the spheroid component.
```
15.5. Objects

nodeComponentSpheroidStandard

<type(abundances)> abundancesGas() Get the abundancesGas property of the spheroid component.

<integer> abundancesGasCount() Compute the count of evolvable quantities in the abundancesGas property of the SpheroidStandard component.

<type(abundances)> abundancesGasIsGettable() Get the abundancesGas property of the spheroid component.

<logical> abundancesGasIsSettable() Specify whether the abundancesGas property of the spheroid component is settable.

<void> abundancesGasRate(<type(abundances)> value) Cumulate to the rate of the abundancesGas property of the SpheroidStandard component.

<void> abundancesGasScale(<type(abundances)> value) Set the scale of the abundancesGas property of the SpheroidStandard component.

<void> abundancesGasSet(<type(abundances)> value) Set the abundancesGas property of the spheroid component.

<type(abundances)> abundancesStellar() Get the abundancesStellar property of the spheroid component.

<integer> abundancesStellarCount() Compute the count of evolvable quantities in the abundancesStellar property of the SpheroidStandard component.

<type(abundances)> abundancesStellarIsGettable() Get the abundancesStellar property of the spheroid component.

<logical> abundancesStellarIsSettable() Specify whether the abundancesStellar property of the spheroid component is settable.

<void> abundancesStellarRate(<type(abundances)> value) Cumulate to the rate of the abundancesStellar property of the SpheroidStandard component.

<void> abundancesStellarScale(<type(abundances)> value) Set the scale of the abundancesStellar property of the SpheroidStandard component.

<void> abundancesStellarSet(<type(abundances)> value) Set the abundancesStellar property of the spheroid component.

<double> angularMomentum() Get the angularMomentum property of the spheroid component.

<integer> angularMomentumCount() Compute the count of evolvable quantities in the angularMomentum property of the SpheroidStandard component.

<double> angularMomentumIsGettable() Get the angularMomentum property of the spheroid component.

<logical> angularMomentumIsSettable() Specify whether the angularMomentum property of the spheroid component is settable.

<void> angularMomentumRate(<double> value) Cumulate to the rate of the angularMomentum property of the SpheroidStandard component.
15. Coding GALACTICUS

<void> angularMomentumScale(<double> value) Set the scale of the angularMomentum property of the SpheroidStandard component.

<void> angularMomentumSet(<double> value) Set the angularMomentum property of the spheroid component.

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<void> createFunctionSet(<function()>) Set the function used to create SpheroidStandard components.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<double> energyGasInput() Get the energyGasInput property of the spheroid component.

<integer> energyGasInputCount() Compute the count of evolvable quantities in the energyGasInput property of the SpheroidStandard component.

<double> energyGasInputIsGettable() Get the energyGasInput property of the spheroid component.

<logical> energyGasInputIsSettable() Specify whether the energyGasInput property of the spheroid component is settable.

<void> energyGasInputRate(<double> value) Cumulate to the rate of the energyGasInput property of the spheroid component.

<void> energyGasInputRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the energyGasInput property of the spheroid component.

<logical> energyGasInputRateIsAttached() Return whether the rate method of the energyGasInput property of the spheroid component has been attached to a function.

<double> halfMassRadius() Get the halfMassRadius property of the spheroid component.

<double> halfMassRadiusIsGettable() Get the halfMassRadius property of the spheroid component.
15.5. Objects

```plaintext
<logical> halfMassRadiusIsSettable() Specify whether the halfMassRadius property of the spheroid component is settable.

<type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<logical> isInitialized() Get the isInitialized property of the spheroid component.

<logical> isInitializedIsGettable() Get the isInitialized property of the spheroid component.

<logical> isInitializedIsSettable() Specify whether the isInitialized property of the spheroid component is settable.

<void> isInitializedSet(<logical> value) Set the isInitialized property of the spheroid component.

<double()> luminositiesStellar() Get the luminositiesStellar property of the spheroid component.

<integer> luminositiesStellarCount() Compute the count of evolvable quantities in the luminositiesStellar property of the SpheroidStandard component.

<double()> luminositiesStellarIsGettable() Get the luminositiesStellar property of the spheroid component.

<logical> luminositiesStellarIsSettable() Specify whether the luminositiesStellar property of the spheroid component is settable.

<void> luminositiesStellarRate(<double()> value) Cumulate to the rate of the luminositiesStellar property of the SpheroidStandard component.

<void> luminositiesStellarScale(<double()> value) Set the scale of the luminositiesStellar property of the SpheroidStandard component.

<void> luminositiesStellarSet(<double()> value) Set the luminositiesStellar property of the spheroid component.

<double> massGas() Get the massGas property of the spheroid component.

<integer> massGasCount() Compute the count of evolvable quantities in the massGas property of the SpheroidStandard component.

<double> massGasIsGettable() Get the massGas property of the spheroid component.

<logical> massGasIsSettable() Specify whether the massGas property of the spheroid component is settable.
```
15. Coding GALACTICUS

```c
<void> massGasRate(<double> value) Cumulate to the rate of the massGas property of the SpheroidStandard component.

<void> massGasScale(<double> value) Set the scale of the massGas property of the SpheroidStandard component.

<void> massGasSet(<double> value) Set the massGas property of the spheroid component.

<double> massGasSink() Get the massGasSink property of the spheroid component.

<integer> massGasSinkCount() Compute the count of evolvable quantities in the massGasSink property of the SpheroidStandard component.

<double> massGasSinkIsGettable() Get the massGasSink property of the spheroid component.

<logical> massGasSinkIsSettable() Specify whether the massGasSink property of the spheroid component is settable.

<void> massGasSinkRate(<double> value) Cumulate to the rate of the massGasSink property of the spheroid component.

<void> massGasSinkRateFunction(<function()> deferredFunction) Set the function to be used for the rate method of the massGasSink property of the spheroid component.

<logical> massGasSinkRateIsAttached() Return whether the rate method of the massGasSink property of the spheroid component has been attached to a function.

<double> massStellar() Get the massStellar property of the spheroid component.

<integer> massStellarCount() Compute the count of evolvable quantities in the massStellar property of the SpheroidStandard component.

<double> massStellarIsGettable() Get the massStellar property of the spheroid component.

<logical> massStellarIsSettable() Specify whether the massStellar property of the spheroid component is settable.

<void> massStellarRate(<double> value) Cumulate to the rate of the massStellar property of the SpheroidStandard component.

<void> massStellarScale(<double> value) Set the scale of the massStellar property of the SpheroidStandard component.

<void> massStellarSet(<double> value) Set the massStellar property of the spheroid component.

<void> nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the spheroid component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.
```
15.5. Objects

<void> output(<integer> integerProperty↔, <integer> integerBufferCount↔, <integer(:,:)>
integerBuffer↔, <integer> doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)>
doubleBuffer↔, <double> time→, <integer> instance→) Generate values of oututable properties.

<void> outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double>
time→, <integer> instance→) Compute a count of outuble properties.

<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 outuble properties.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→,
<logical> [haloLoaded]→) Compute the gravitational potential.

<double> radius() Get the radius property of the spheroid component.

<logical> radiusIsGettable() Get the radius property of the spheroid component.

<void> radiusSet(<double> value) Specify whether the radius property of the spheroid component is settable.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→,
<logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→,
<logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<logical> standardIsActive() Return whether the standard implementation of the spheroid component class is active.

<type(history)> starFormationHistory() Get the starFormationHistory property of the spheroid component.

<integer> starFormationHistoryCount() Compute the count of evolvable quantities in the starFormationHistory
property of the SpheroidStandard component.

<type(history)> starFormationHistoryIsGettable() Get the starFormationHistory property of
the spheroid component.

<logical> starFormationHistoryIsSettable() Specify whether the starFormationHistory property of the spheroid component is settable.
15. Coding Galacticus

<void> starFormationHistoryRate(<type(history)> value)Cumulate to the rate of the starFormationHistory property of the SpheroidStandard component.

<void> starFormationHistoryScale(<type(history)> value) Set the scale of the starFormationHistory property of the SpheroidStandard component.

<void> starFormationHistorySet(<type(history)> value) Set the starFormationHistory property of the spheroid component.

<double> starFormationRate() Get the starFormationRate property of the spheroid component.

<void> starFormationRateFunction(<function()> deferredFunction) Set the function to be used for the get method of the starFormationRate property of the spheroid component.

<logical> starFormationRateIsAttached() Return whether the get method of the starFormationRate property of the spheroid component has been attached to a function.

<double> starFormationRateIsGettable() Get the starFormationRate property of the spheroid component.

<logical> starFormationRateIsSettable() Specify whether the starFormationRate property of the spheroid component is settable.

<type(history)> stellarPropertiesHistory() Get the stellarPropertiesHistory property of the spheroid component.

<integer> stellarPropertiesHistoryCount() Compute the count of evolvable quantities in the stellarPropertiesHistory property of the SpheroidStandard component.

<type(history)> stellarPropertiesHistoryIsGettable() Get the stellarPropertiesHistory property of the spheroid component.

<logical> stellarPropertiesHistoryIsSettable() Specify whether the stellarPropertiesHistory property of the spheroid component is settable.

<void> stellarPropertiesHistoryRate(<type(history)> value) Cumulate to the rate of the stellarPropertiesHistory property of the SpheroidStandard component.

<void> stellarPropertiesHistoryScale(<type(history)> value) Set the scale of the stellarPropertiesHistory property of the SpheroidStandard component.

<void> stellarPropertiesHistorySet(<type(history)> value) Set the stellarPropertiesHistory property of the spheroid component.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

<double> velocity() Get the velocity property of the spheroid component.

<double> velocityIsGettable() Get the velocity property of the spheroid component.

<logical> velocityIsSettable() Specify whether the velocity property of the spheroid component is settable.

<void> velocitySet(<double> value) Set the velocity property of the spheroid component.
15.5. Objects

nodeComponentSpin

<void> builder(<type(node)>componentDefinition) Build a nodeComponent from a supplied XML definition.

<double> 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) Deserializethe evolvable quantities from an array.

<void> destroy() Destroy the object.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius, <componentType> [componentType], <massType> [massType], <weightBy> [weightBy], <integer> [weightIndex], <logical> [haloLoaded]) Compute the mass enclosed within a radius.

<type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulatetotherateofthe hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulatetotherateofthe hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count, <varying_string>name) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the spin component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty, <integer> integerBufferCount, <integer(:,:)integerBuffer, <integer> doubleProperty, <integer> doubleBufferCount, <double(:,:)doubleBuffer, <double> time, <integer> instance) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount, <integer> doublePropertyCount, <double> time, <integer> instance) Compute a count of outputtable properties.
outputNames(<integer> integerProperty ↔, <char[∗](:)> integerPropertyNames ↔, <char[∗](:)> integerPropertyComments ↔, <double(:)> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char[∗](:)> doublePropertyNames ↔, <char[∗](:)> doublePropertyComments ↔, <double(:)> doublePropertyUnitsSI ↔, <double> time →, <integer> instance →) Generate names of output-puttable properties.

potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

presetIsActive() Return whether the preset implementation of the spin component class is active.

randomIsActive() Return whether the random implementation of the spin component class is active.

rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

serializeCount() Return a count of the number of evolvable quantities to be evolved.

serializeRates(<double(:)> array →) Serialize the evolvable rates to an array.

serializeScales(<double(:)> array →) Serialize the evolvable scales to an array.

serializeValues(<double(:)> array →) Serialize the evolvable quantities to an array.

spin() Get the spin property of the Spin component.

spinCount() Compute the count of evolvable quantities in the spin property of the SpinRandom component.

spinGrowthRate() Get the spinGrowthRate property of the spin component.

spinGrowthRateIsGettable() Get the spinGrowthRate property of the spin component.

spinGrowthRateIsSettable() Specify whether the spinGrowthRate property of the spin component is settable.

spinGrowthRateSet(<double> value) Set the spinGrowthRate property of the spin component.

spinIsGettable() Get the spin property of the spin component.

spinIsSettable() Specify whether the spin property of the spin component is settable.

spinRate(<double> value) Cumulate to the rate of the spin property of the SpinRandom component.

spinScale(<double> value) Set the scale of the spin property of the SpinRandom component.

spinSet(<double> value) Set the spin property of the spin component.

surfaceDensity(<double(3)> positionCylindrical →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the surface density.

type(varying_string) type() Return the type of this object.
nodeComponentSpinNull

```c
<void> builder(<*type(node)> componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<*type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count→, <varying_string> name←) Return the name of a property given its index.

<logical> nullIsActive() Return whether the null implementation of the spin component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty←, <integer> integerBufferCount←, <integer(:,:)> integerBuffer←, <integer> doubleProperty←, <integer> doubleBufferCount←, <double(:,:)> doubleBuffer←, <double> time→, <integer> instance→) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount←, <integer> doublePropertyCount←, <double> time→, <integer> instance→) Compute a count of outputtable properties.
```
15. Coding Galacticus

```c
<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.

<double> potential(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the gravitational potential.

<logical> presetIsActive() Return whether the preset implementation of the spin component class is active.

<logical> randomIsActive() Return whether the random implementation of the spin component class is active.

<void> readRaw(<integer> fileHandle →) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius →, <componentType> [componentType] →, <massType> [massType] →, <logical> [haloLoaded] →) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double()> array →) Serialize the evolvable rates to an array.

<void> serializeScales(<double()> array →) Serialize the evolvable scales to an array.

<void> serializeValues(<double()> array →) Serialize the evolvable quantities to an array.

<double> spin() Get the spin property of the spin component.

<integer> spinCount() Compute the count of evolvable quantities in the spin property of the SpinRandom component.

<double> spinGrowthRate() Get the spinGrowthRate property of the spin component.

<double> spinGrowthRateIsGettable() Get the spinGrowthRate property of the spin component.

<logical> spinGrowthRateIsSettable() Specify whether the spinGrowthRate property of the spin component is settable.

<void> spinGrowthRateSet(<double> value) Set the spinGrowthRate property of the spin component.

<double> spinIsGettable() Get the spin property of the spin component.

<logical> spinIsSettable() Specify whether the spin property of the spin component is settable.

<void> spinRate(<double> value) Cumulate to the rate of the spin property of the SpinRandom component.

<void> spinScale(<double> value) Set the scale of the spin property of the SpinRandom component.

<void> spinSet(<double> value) Set the spin property of the spin component.
```
15.5. Objects

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

cnodeComponentSpinPreset
<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<*type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the spin component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.

<void> output(<integer> integerProperty→, <integer> integerBufferCount↔, <integer(:,:)> integerBuffer↔, <integer>doubleProperty↔, <integer> doubleBufferCount↔, <double(:,:)> doubleBuffer↔, <double> time→, <integer> instance→) Generate values of outputtable properties.
15. Coding GALACTICUS

```c
<void> outputCount(<integer> integerPropertyCount↔, <integer> doublePropertyCount↔, <double> time→, <integer> instance→) Compute a count of outputtable properties.

<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.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the gravitational potential.

<logical> presetIsActive() Return whether the preset implementation of the spin component class is active.

<logical> randomIsActive() Return whether the random implementation of the spin component class is active.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<double> spin() Get the spin property of the spin component.

<integer> spinCount() Compute the count of evolvable quantities in the spin property of the SpinRandom component.

<double> spinGrowthRate() Get the spinGrowthRate property of the spin component.

<double> spinGrowthRateIsGettable() Get the spinGrowthRate property of the spin component.

<logical> spinGrowthRateIsSettable() Specify whether the spinGrowthRate property of the spin component is settable.

<void> spinGrowthRateSet(<double> value) Set the spinGrowthRate property of the spin component.

<double> spinIsGettable() Get the spin property of the spin component.

<logical> spinIsSettable() Specify whether the spin property of the spin component is settable.

<void> spinRate(<double> value) Cumulate to the rate of the spin property of the SpinRandom component.
```
15.5. Objects

<void> spinScale(<double> value) Set the scale of the spin property of the SpinRandom component.

<void> spinSet(<double> value) Set the spin property of the spin component.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

nodeComponentSpinRandom

<void> builder(<*type(node)>componentDefinition→) Build a nodeComponent from a supplied XML definition.

<double> 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.

<void> dump() Generate an ASCII dump of all properties.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all properties.

<double> enclosedMass(<double> radius→, <componentType> [componentType]→, <massType> [massType]→, <weightBy> [weightBy]→, <integer> [weightIndex]→, <logical> [haloLoaded]→) Compute the mass enclosed within a radius.

<*type(treeNode)> host() Return a pointer to the host treeNode object.

<void> hotHaloCoolingAbundancesRate(<type(abundances)> value) Cumulate to the rate of the hotHaloCoolingAbundances property of the hotHalo component.

<void> hotHaloCoolingAngularMomentumRate(<double> value) Cumulate to the rate of the hotHaloCoolingAngularMomentum property of the hotHalo component.

<void> hotHaloCoolingMassRate(<double> value) Cumulate to the rate of the hotHaloCoolingMass property of the hotHalo component.

<void> initialize() Initialize the object.

<void> nameFromIndex(<integer> count→, <varying_string>name←) Return the name of a property given is index.

<logical> nullIsActive() Return whether the null implementation of the spin component class is active.

<void> odeStepRatesInitialize() Initialize rates for evolvable properties.

<void> odeStepScalesInitialize() Initialize scales for evolvable properties.
<void> output(<integer> integerProperty→, <integer> integerBufferCount→, <integer(:,:)>
integerBuffer→, <integer>doubleProperty→, <integer> doubleBufferCount→, <double(:,:)>
doubleBuffer→, <double> time→, <integer> instance→) Generate values of outputtable properties.

<void> outputCount(<integer> integerPropertyCount→, <integer> doublePropertyCount→, <double>
time→, <integer> instance→) Compute a count of outputtable properties.

<void> outputNames(<integer> integerProperty→, <char[*](::)> integerPropertyNames→, <char[*](::)> integerPropertyComments→, <integer> doubleProperty→,
<double(:)> integerPropertyUnitsSI→, <integer> doubleProperty→,
<char[*](::)> doublePropertyNames→, <char[*](::)> doublePropertyComments→, <double(:)> doublePropertyUnitsSI→, <double> time→, <integer> instance→) Generate names of outputtable properties.

<double> potential(<double> radius→, <componentType> [componentType]→, <massType> [massType]→,
<logical> [haloLoaded]→) Compute the gravitational potential.

<logical> presetIsActive() Return whether the preset implementation of the spin component class is active.

<logical> randomIsActive() Return whether the random implementation of the spin component class is active.

<void> readRaw(<integer> fileHandle→) Read a binary dump of the nodeComponent from the given fileHandle.

<double> rotationCurve(<double> radius→, <componentType> [componentType]→, <massType> [massType]→,
<logical> [haloLoaded]→) Compute the rotation curve.

<double> rotationCurveGradient(<double> radius→, <componentType> [componentType]→, <massType> [massType]→,
<logical> [haloLoaded]→) Compute the rotation curve gradient.

<integer> serializeCount() Return a count of the number of evolvable quantities to be evolved.

<void> serializeRates(<double(:)> array→) Serialize the evolvable rates to an array.

<void> serializeScales(<double(:)> array→) Serialize the evolvable scales to an array.

<void> serializeValues(<double(:)> array→) Serialize the evolvable quantities to an array.

<double> spin() Get the spin property of the spin component.

<integer> spinCount() Compute the count of evolvable quantities in the spin property of the SpinRandom component.

<double> spinGrowthRate() Get the spinGrowthRate property of the spin component.

<double> spinGrowthRateIsGettable() Get the spinGrowthRate property of the spin component.

<logical> spinGrowthRateIsSettable() Specify whether the spinGrowthRate property of the spin component is settable.

<void> spinGrowthRateSet(<double> value) Set the spinGrowthRate property of the spin component.

<double> spinIsGettable() Get the spin property of the spin component.
15.5. Objects

<logical> spinIsSettable() Specify whether the spin property of the spin component is settable.

<void> spinRate(<double> value) Cumulate to the rate of the spin property of the SpinRandom component.

<void> spinScale(<double> value) Set the scale of the spin property of the SpinRandom component.

<void> spinSet(<double> value) Set the spin property of the spin component.

<double> surfaceDensity(<double(3)> positionCylindrical→, <componentType> [componentType]→, <massType> [massType]→, <logical> [haloLoaded]→) Compute the surface density.

<type(varying_string)> type() Return the type of this object.

radiationStructure

<void> define(<integer(:)> radiationTypes→) Define the radiation components active in a given radiation object.

<double> 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.

<double> 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.

<logical> isDefined() Return true if the radiation component is defined, false otherwise.

<void> set(<type(treeNode)>thisNode) Set the radiation components in the radiation object.

<double> temperature(<integer(:)> [radiationType]→) Return the temperature (in units of Kelvin) of the given radiation structure.

<double> time() The cosmic time at which this radiation object was set.

rootFinder

<double> find(<double> [rootGuess]|<double(2)> [rootRange]) Find the root of the function given an initial guess or range.

<logical> isInitialized() Return the initialization state of a rootFinder object.

<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.

<void> rootFunction(<function(<double> x→) rootFunction) Set the function that evaluates $f(x)$ to use in a rootFinder object.

<void> tolerance(<double> [toleranceAbsolute]→, <double> [toleranceRelative]→) Set the tolerance to use in a rootFinder object.

<void> type(<type(fgsl_root_fsolver_type)> solverType→) Set the type of algorithm to use in a rootFinder object.
table

<void> destroy() Destroy the table.

table1D

<void> destroy() Destroy the table.

<double> interpolate(<double> x,<integer> [table]) Interpolate to x in the table\textsuperscript{th} table.

<double> interpolateGradient(<double> x,<integer> [table]) Interpolate the gradient to x in the table\textsuperscript{th} table.

<logical> isMonotonic(<enumeration> [directionDecreasing|directionIncreasing],<logical> [allowEqual],<integer> [table]) Return true if the table y-values are monotonic. Optionally, the direction of monotonicity can be specified via the 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 allowEqual argument. If table is specified then the table\textsuperscript{th} table is used for the y-values, otherwise the first table is used.

<void> reverse(<type(table)> reversedSelf,<integer> [table]) Reverse the table (i.e. swap x and y components) and return in reversedSelf. If table is specified then the table\textsuperscript{th} table is used for the y-values, otherwise the first table is used.

<integer> size() Return the size (i.e. number of x-values) in the table.

<double> x(<integer> i) Return the i\textsuperscript{th} x-value.

<double> xEffective(<double> x) Return the effective value of x to use in table interpolations.

<double(:)> xs(<integer> i) Return an array of all x-values.

<double> y(<integer> i,<integer> [table]) Return the i\textsuperscript{th} y-value. If table is specified then the table\textsuperscript{th} table is used for the y-values, otherwise the first table is used.

<double(:)> ys(<integer> i,<integer> [table]) Return an array of all y-values. If table is specified then the table\textsuperscript{th} table is used for the y-values, otherwise the first table is used.

table1DGeneric

<void> create(<double(:)> x,<integer> [tableCount]) Create the object with the specified x values, and with tableCount tables.

<void> destroy() Destroy the table.

<double> interpolate(<double> x,<integer> [table]) Interpolate to x in the table\textsuperscript{th} table.

<double> interpolateGradient(<double> x,<integer> [table]) Interpolate the gradient to x in the table\textsuperscript{th} table.

<logical> isMonotonic(<enumeration> [directionDecreasing|directionIncreasing],<logical> [allowEqual],<integer> [table]) Return true if the table y-values are monotonic. Optionally, the direction of monotonicity can be specified via the 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 allowEqual argument. If table is specified then the table\textsuperscript{th} table is used for the y-values, otherwise the first table is used.
15.5. Objects

`<void> populate(<double>|<double(:)> y,<integer> [i],<integer> [table])` Populate the $table^{th}$ table with elements $y$. If $y$ is a scalar, then the index, $i$, of the element to set must also be specified.

`<void> reverse(<type(table)> reversedSelf,<integer> [table])` Reverse the table (i.e. swap $x$ and $y$ components) and return in `reversedSelf`. If `table` is specified then the $table^{th}$ table is used for the $y$-values, otherwise the first table is used.

`<integer> size()` Return the size (i.e. number of $x$-values) in the table.

`<double> x(<integer> i)` Return the $i^{th}$ $x$-value.

`<double> xEffective(<double> x)` Return the effective value of $x$ to use in table interpolations.

`<double(:)> xs(<integer> i)` Return an array of all $x$-values.

`<double> y(<integer> i,<integer> [table])` Return the $i^{th}$ $y$-value. If `table` is specified then the $table^{th}$ table is used for the $y$-values, otherwise the first table is used.

`<double(:)> ys(<integer> i,<integer> [table])` Return an array of all $y$-values. If `table` is specified then the $table^{th}$ table is used for the $y$-values, otherwise the first table is used.

**table1DLinearCSpline**

`<void> create(<double> xMinimum,<double> xMaximum,<integer>xCount,<integer>[tableCount])` Create the object with $x$-values spanning the range $xMinimum$ to $xMaximum$ in $xCount$ steps, and with $tableCount$ tables.

`<void> destroy()` Destroy the table.

`<double> interpolate(<double> x,<integer> [table])` Interpolate to $x$ in the $table^{th}$ table.

`<double> interpolateGradient(<double> x,<integer> [table])` Interpolate the gradient to $x$ in the $table^{th}$ table.

`<logical> isMonotonic(<enumeration> [directionDecreasing|directionIncreasing],<logical> [allowEqual],<integer> [table])` Return true if the table $y$-values are monotonic. Optionally, the direction of monotonicity can be specified via the `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 `allowEqual` argument. If `table` is specified then the $table^{th}$ table is used for the $y$-values, otherwise the first table is used.

`<void> populate(<double>|<double(:)> y,<integer> [i],<integer> [table])` Populate the $table^{th}$ table with elements $y$. If $y$ is a scalar, then the index, $i$, of the element to set must also be specified.

`<void> reverse(<type(table)> reversedSelf,<integer> [table])` Reverse the table (i.e. swap $x$ and $y$ components) and return in `reversedSelf`. If `table` is specified then the $table^{th}$ table is used for the $y$-values, otherwise the first table is used.

`<integer> size()` Return the size (i.e. number of $x$-values) in the table.

`<double> x(<integer> i)` Return the $i^{th}$ $x$-value.

`<double> xEffective(<double> x)` Return the effective value of $x$ to use in table interpolations.

`<double(:)> xs(<integer> i)` Return an array of all $x$-values.
15. Coding GALACTICUS

\begin{verbatim}
<double> y(<integer> i,<integer> [table]) Return the \(i^{th}\) \(y\)-value. If \texttt{table} is specified then the \(table^{th}\) table is used for the \(y\)-values, otherwise the first table is used.

<double(:)> ys(<integer> i,<integer> [table]) Return an array of all \(y\)-values. If \texttt{table} is specified then the \(table^{th}\) table is used for the \(y\)-values, otherwise the first table is used.

table1DLinearLinear
<void> create(<double> xMinimum,<double> xMaximum,<integer>xCount,<integer>[tableCount])
Create the object with \(x\)-values spanning the range \(xMinimum\) to \(xMaximum\) in \(xCount\) steps, and with \(tableCount\) tables.

<void> destroy() Destroy the table.

<double> interpolate(<double> x,<integer> [table]) Interpolate to \(x\) in the \(table^{th}\) table.

<double> interpolateGradient(<double> x,<integer> [table]) Interpolate the gradient to \(x\) in the \(table^{th}\) table.

<logical> isMonotonic(<enumeration> [directionDecreasing|directionIncreasing],<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 \(table^{th}\) table is used for the \(y\)-values, otherwise the first table is used.

<void> populate(<double>|<double(:)> y,<integer> [i],<integer> [table]) Populate the \(table^{th}\) table with elements \(y\). If \(y\) is a scalar, then the index, \(i\), of the element to set must also be specified.

<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 \(table^{th}\) table is used for the \(y\)-values, otherwise the first table is used.

<integer> size() Return the size (i.e. number of \(x\)-values) in the table.

<double> x(<integer> i) Return the \(i^{th}\) \(x\)-value.

<double> xEffective(<double> x) Return the effective value of \(x\) to use in table interpolations.

<double(:)> xs(<integer> i) Return an array of all \(x\)-values.

<double> y(<integer> i,<integer> [table]) Return the \(i^{th}\) \(y\)-value. If \texttt{table} is specified then the \(table^{th}\) table is used for the \(y\)-values, otherwise the first table is used.

<double(:)> ys(<integer> i,<integer> [table]) Return an array of all \(y\)-values. If \texttt{table} is specified then the \(table^{th}\) table is used for the \(y\)-values, otherwise the first table is used.

table1DLogarithmicCSpline
<void> create(<double> xMinimum,<double> xMaximum,<integer>xCount,<integer>[tableCount])
Create the object with \(x\)-values spanning the range \(xMinimum\) to \(xMaximum\) in \(xCount\) steps, and with \(tableCount\) tables.

<void> destroy() Destroy the table.
\end{verbatim}
15.5. Objects

<double> interpolate(<double> x, <integer> [table]) Interpolate to x in the $table^\text{th}$ table.

<double> interpolateGradient(<double> x, <integer> [table]) Interpolate the gradient to x in the $table^\text{th}$ table.

<logical> isMonotonic(<enumeration> [directionDecreasing|directionIncreasing], <logical> [allowEqual], <integer> [table]) Return true if the table y-values are monotonic. Optionally, the direction of monotonicity can be specified via the 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 allowEqual argument. If table is specified then the $table^\text{th}$ table is used for the y-values, otherwise the first table is used.

<void> populate(<double>|<double(:)> y, <integer> [i], <integer> [table]) Populate the $table^\text{th}$ table with elements y. If y is a scalar, then the index, i, of the element to set must also be specified.

<void> reverse(<type(table)> reversedSelf, <integer> [table]) Reverse the table (i.e. swap x and y components) and return in reversedSelf. If table is specified then the $table^\text{th}$ table is used for the y-values, otherwise the first table is used.

<integer> size() Return the size (i.e. number of x-values) in the table.

<double> x(<integer> i) Return the $i^{\text{th}}$ x-value.

<double(:)> xs(<integer> i) Return an array of all x-values.

<double> y(<integer> i, <integer> [table]) Return the $i^{\text{th}}$ y-value. If table is specified then the $table^\text{th}$ table is used for the y-values, otherwise the first table is used.

<double(:)> ys(<integer> i, <integer> [table]) Return an array of all y-values. If table is specified then the $table^\text{th}$ table is used for the y-values, otherwise the first table is used.

table1DLogarithmicLinear

<void> create(<double> xMinimum, <double> xMaximum, <integer> xCount, <integer> [tableCount]) Create the object with x-values spanning the range $x_{\text{Minimum}}$ to $x_{\text{Maximum}}$ in $x_{\text{Count}}$ steps, and with tableCount tables.

<void> destroy() Destroy the table.

<double> interpolate(<double> x, <integer> [table]) Interpolate to x in the $table^\text{th}$ table.

<double> interpolateGradient(<double> x, <integer> [table]) Interpolate the gradient to x in the $table^\text{th}$ table.

<logical> isMonotonic(<enumeration> [directionDecreasing|directionIncreasing], <logical> [allowEqual], <integer> [table]) Return true if the table y-values are monotonic. Optionally, the direction of monotonicity can be specified via the 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 allowEqual argument. If table is specified then the $table^\text{th}$ table is used for the y-values, otherwise the first table is used.

<void> populate(<double>|<double(:)> y, <integer> [i], <integer> [table]) Populate the $table^\text{th}$ table with elements y. If y is a scalar, then the index, i, of the element to set must also be specified.
15. Coding Galacticus

```c
<void> reverse(<type(table)> reversedSelf, <integer> [table]) Reverse the table (i.e. swap x and y components) and return in reversedSelf. If table is specified then the table\textsuperscript{th} table is used for the y-values, otherwise the first table is used.

<integer> size() Return the size (i.e. number of x-values) in the table.

<double> x(<integer> i) Return the \(i\)th x-value.

<double> xEffective(double x) Return the effective value of x to use in table interpolations.

<double(:)> xs(<integer> i) Return an array of all x-values.

<double> y(<integer> i, <integer> [table]) Return the \(i\)th y-value. If table is specified then the table\textsuperscript{th} table is used for the y-values, otherwise the first table is used.

<double(:)> ys(<integer> i, <integer> [table]) Return an array of all y-values. If table is specified then the table\textsuperscript{th} table is used for the y-values, otherwise the first table is used.

treeNode

<class(nodeComponentBasic)> basic(<integer> [instance] \rightarrow, <logical> [autoCreate] \rightarrow) 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.

<integer> basicCount() Returns the number of basic components in the node.

<void> basicCreate(<class(nodeComponentBasic)> [template] \rightarrow) Create a basic component in the node. If no template is specified use the active implementation of this class.

<void> basicDestroy() Destroy the basic component(s) of the node.

<void> basicMove(<type(treeNode>) targetNode \leftrightarrow) HASH(0x191d5490)

<void> basicRemove(<integer> [instance] \rightarrow) 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.

<class(nodeComponentBlackHole)> blackHole(<integer> [instance] \rightarrow, <logical> [autoCreate] \rightarrow) 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.

<integer> blackHoleCount() Returns the number of blackHole components in the node.

<void> blackHoleCreate(<class(nodeComponentBlackHole)> [template] \rightarrow) Create a blackHole component in the node. If no template is specified use the active implementation of this class.

<void> blackHoleDestroy() Destroy the blackHole component(s) of the node.

<void> blackHoleMove(<type(treeNode>) targetNode \leftrightarrow) HASH(0x191d8f70)

<void> blackHoleRemove(<integer> [instance] \rightarrow) 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.

442
15.5. Objects

<void> componentBuilder(<*type(node)> nodeDefinition→) Build components in this node given an XML description of their properties.

<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.

<*type(nodeEvent)> createEvent() Create a nodeEvent object in this node.

<*class(nodeComponentDarkMatterProfile)> 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.

<integer> darkMatterProfileCount() Returns the number of darkMatterProfile components in the node.

<void> darkMatterProfileCreate(<class(nodeComponentDarkMatterProfile)> [template]→) Create a darkMatterProfile component in the node. If no template is specified use the active implementation of this class.

<void> darkMatterProfileDestroy() Destroy the darkMatterProfile component(s) of the node.

<void> darkMatterProfileMove(<type(treeNode)> targetNode↔) HASH(0x191d9260)

<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.

<void> deserializeRates(<double(:)> array→) Deserialize rates from array.

<void> deserializeScales(<double(:)> array→) Deserialize scales from array.

<void> deserializeValues(<double(:)> array→) Deserialize values from array.

<void> destroy() Destroy this node.

<*class(nodeComponentDisk)> 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.

<integer> diskCount() Returns the number of disk components in the node.

<void> diskCreate(<class(nodeComponentDisk)> [template]→) Create a disk component in the node. If no template is specified use the active implementation of this class.

<void> diskDestroy() Destroy the disk component(s) of the node.

<void> diskMove(<type(treeNode)> targetNode↔) HASH(0x191d86a0)

<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.

<void> dump() Generate an ASCII dump of all content of a node.

<void> dumpRaw(<integer> fileHandle→) Generate a binary dump of all content of a node.
15. Coding Galacticus

```c
<*>type(treeNode)> earliestProgenitor() Return a pointer to the earliest progenitor (along the main branch) of this node.

<*>class(nodeComponentFormationTime)> 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.

<*>class(nodeComponentHotHalo)> hotHalo(<integer> [instance] →, <logical> [autoCreate] →) 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.

<*>class(nodeComponentIndices)> indices(<integer> [instance] →, <logical> [autoCreate] →) 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.

<*>type(treeNode)> index() Return the index of this node.

<*>class(nodeComponentIndices)> indicesCount() Returns the number of indices components in the node.

<*>class(nodeComponentFormationTime)> formationTimeCount() Returns the number of formationTime components in the node.

<*>class(nodeComponentHotHalo)> hotHaloCount() Returns the number of hotHalo components in the node.

<*>class(nodeComponentIndices)> indicesCreate(<class(nodeComponentIndices)> [template] →) Create a formationTime component in the node. If no template is specified use the active implementation of this class.

<*>class(nodeComponentFormationTime)> formationTimeCreate(<class(nodeComponentFormationTime)> [template] →) Create a formationTime component in the node. If no template is specified use the active implementation of this class.

<*>class(nodeComponentHotHalo)> hotHaloCreate(<class(nodeComponentHotHalo)> [template] →) Create a hotHalo component in the node. If no template is specified use the active implementation of this class.

<*>class(nodeComponentIndices)> indicesCreate(<class(nodeComponentIndices)> [template] →) Create a hotHalo component in the node. If no template is specified use the active implementation of this class.

<*>class(nodeComponentIndices)> indicesDestroy() Destroy the indices component(s) of the node.

<*>class(nodeComponentFormationTime)> formationTimeDestroy() Destroy the formationTime component(s) of the node.

<*>class(nodeComponentHotHalo)> hotHaloDestroy() Destroy the hotHalo component(s) of the node.

<*>class(nodeComponentIndices)> indicesDestroy() Destroy the indices component(s) of the node.

<*>class(nodeComponentFormationTime)> formationTimeMove(<type(treeNode)> targetNode ↔) HASH(0x191d8990)

<*>class(nodeComponentHotHalo)> hotHaloMove(<type(treeNode)> targetNode ↔) HASH(0x191c3100)

<*>class(nodeComponentIndices)> indicesMove(<type(treeNode)> targetNode ↔) HASH(0x191dee60)

<*>class(nodeComponentIndices)> indicesRemove(<integer> [instance] →) 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.

<*>class(nodeComponentHotHalo)> hotHaloRemove(<integer> [instance] →) 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.

<*>class(nodeComponentIndices)> indicesRemove(<integer> [instance] →) Remove an instance of the indices component, shifting other instances to keep the array contiguous. If no instance is specified, the first instance is assumed.

<*>class(nodeComponentFormationTime)> formationTimeRemove(<integer> [instance] →) 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.

<*>class(nodeComponentHotHalo)> hotHaloRemove(<integer> [instance] →) 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.

<*>class(nodeComponentIndices)> indicesRemove(<integer> [instance] →) Remove an instance of the indices component, shifting other instances to keep the array contiguous. If no instance is specified, the first instance is assumed.
```

444
15.5. Objects

`<void> indicesRemove(<integer> [instance]→)` Remove an instance of the indices component, shifting other instances to keep the array contiguous. If no `instance` is specified, the first instance is assumed.

`<void> initialize(<integer(kind_int8)> index→)` Initialize this node (assigns a unique identifier, creates generic components).

`<*class(nodeComponentInterOutput)> interOutput(<integer> [instance]→, <logical> [autoCreate]→)` Return a interOutput 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.

`<integer> interOutputCount()` Returns the number of interOutput components in the node.

`<void> interOutputCreate(<class(nodeComponentInterOutput)> [template]→)` Create a interOutput component in the node. If no `template` is specified use the active implementation of this class.

`<void> interOutputDestroy()` Destroy the interOutput component(s) of the node.

`<void> interOutputMove(<type(treeNode)> targetNode↔)` HASH(0x191c2e10)

`<void> interOutputRemove(<integer> [instance]→)` Remove an instance of the interOutput component, shifting other instances to keep the array contiguous. If no `instance` is specified, the first instance is assumed.

`<logical> isOnMainBranch()` Return true if this node is on the main branch of its tree, false otherwise.

`<logical> isPrimaryProgenitor()` Return true if this node is the primary progenitor of its descendent, false otherwise.

`<logical> isPrimaryProgenitorOf(<integer(kind_int8)> targetNodeIndex→,<*type(treeNode)> targetNode→)` Return true if this node is the primary progenitor of the specified (by index or pointer) node, false otherwise.

`<logical> isSatellite()` Return true if this node is a satellite, false otherwise.

`<*type(treeNode)> lastSatellite()` Return a pointer to the last satellite in the list of satellites belonging to this node.

`<double> mapDouble0(<*function()> mapFunction)` Map a scalar double function over components.

`<void> mapVoid(<*function()> mapFunction)` Map a void function over components.

`<*type(treeNode)> mergesWith()` Return a pointer to the node with which this node will merge.

`<*class(nodeComponentMergingStatistics)> mergingStatistics(<integer> [instance]→, <logical> [autoCreate]→)` 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.

`<integer> mergingStatisticsCount()` Returns the number of mergingStatistics components in the node.

`<void> mergingStatisticsCreate(<class(nodeComponentMergingStatistics)> [template]→)` Create a mergingStatistics component in the node. If no `template` is specified use the active implementation of this class.
15. Coding GALACTICUS

<void> mergingStatisticsDestroy() Destroy the mergingStatistics component(s) of the node.

<void> mergingStatisticsMove(<type(treeNode)> targetNode) HASH(0x191df440)

<void> mergingStatisticsRemove(<integer> [instance] →) Remove an instance of the mergingStatistics component, shifting other instances to keep the array contiguous. If no instance is specified, the first instance is assumed.

<void> moveComponentsTo(<class(treeNode)> targetNode) Move components from a node to targetNode.

<varying_string> nameFromIndex(<integer> index →) Return the name of a property given its index in a node.

<void> odeStepRatesInitialize() Initialize rates of evolvable properties.

<void> odeStepScalesInitialize() Initialize tolerance scales of evolvable properties.

<void> output(<integer> integerProperty ↔, <integer> integerBufferCount ↔, <integer(:,:)> integerBuffer ↔, <integer>doubleProperty ↔, <integer> doubleBufferCount ↔, <double(:,:)> doubleBuffer ↔, <double> time →) Populate output buffers with properties for a node.

<void> outputCount(<integer> integerPropertyCount ↔, <integer> doublePropertyCount ↔, <double> time →) Increment the count of properties to output for a node.

<void> outputNames(<integer> integerProperty ↔, <char[*](:)> integerPropertyNames ↔, <char[*](:)> integerPropertyComments ↔, <double(:)> integerPropertyUnitsSI ↔, <integer> doubleProperty ↔, <char[*](:)> doublePropertyNames ↔, <char[*](:)> doublePropertyComments ↔, <double(:)> doublePropertyUnitsSI ↔, <double> time →) Establish the names of properties to output for a node.

<class(nodeComponentPosition)> position(<integer> [instance] →, <logical> [autoCreate] →) Return a position 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.

<integer> positionCount() Returns the number of position components in the node.

<void> positionCreate(<class(nodeComponentPosition)> [template] →) Create a position component in the node. If no template is specified use the active implementation of this class.

<void> positionDestroy() Destroy the position component(s) of the node.

<void> positionMove(<type(treeNode)> targetNode) HASH(0x191d8c80)

<void> positionRemove(<integer> [instance] →) Remove an instance of the position component, shifting other instances to keep the array contiguous. If no instance is specified, the first instance is assumed.

<void> readRaw(<integer> fileHandle →) Read a binary dump of all content of a node.

<void> removeFromHost() Remove this node from the satellite population of its host halo.

<void> removeFromMergee() Remove this node from the list of mergees associated with its merge target.
<void> removePairedEvent(<type(nodeEvent)> event→) Remove a paired nodeEvent from this node.

<class(nodeComponentSatellite)> satellite(<integer> [instance]→, <logical> [autoCreate]→)
Return a satellite 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.

<integer> satelliteCount() Returns the number of satellite components in the node.

<void> satelliteCreate(<class(nodeComponentSatellite)> [template]→) Create a satellite component in the node. If no template is specified use the active implementation of this class.

<void> satelliteDestroy() Destroy the satellite component(s) of the node.

<void> satelliteMove(<type(treeNode)> targetNode↔) HASH(0x191df150)

<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.

<integer> serializeCount() Return a count of the number of evolvable properties of the serialized object.

<void> serializeRates(<double[::]> array←) Serialize rates to array.

<void> serializeScales(<double[::]> array←) Serialize scales to array.

<void> serializeValues(<double[::]> array←) Serialize values to array.

<class(nodeComponentSpheroid)> 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.

<integer> spheroidCount() Returns the number of spheroid components in the node.

<void> spheroidCreate(<class(nodeComponentSpheroid)> [template]→) Create a spheroid component in the node. If no template is specified use the active implementation of this class.

<void> spheroidDestroy() Destroy the spheroid component(s) of the node.

<void> spheroidMove(<type(treeNode)> targetNode↔) HASH(0x191d5780)

<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.

<class(nodeComponentSpin)> 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.

<integer> spinCount() Returns the number of spin components in the node.

<void> spinCreate(<class(nodeComponentSpin)> [template]→) Create a spin component in the node. If no template is specified use the active implementation of this class.

<void> spinDestroy() Destroy the spin component(s) of the node.
15. Coding GALACTICUS

```c
<void> spinMove(<type(treeNode)> targetNode ↔) HASH(0x191c33f0)

<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.

<type(varying_string)> type() Return the type of this node.

<integer(kind_int8)> uniqueID() Return the unique identifier for this node.

<void> uniqueIDSet(<integer(kind_int8)> uniqueID →) Set the unique identifier for this node.

<void> walkBranch(<*type(treeNode)> startNode ↔, <*type(treeNode)> nextNode ↔) Return a pointer to the next node when performing a walk of a single branch of the tree, excluding satellites.

<void> walkBranchWithSatellites(<*type(treeNode)> startNode ↔, <*type(treeNode)> nextNode ↔) Return a pointer to the next node when performing a walk of a single branch of the tree, including satellites.

<void> walkTree(<*type(treeNode)> nextNode ↔) Return a pointer to the next node when performing a walk of the entire tree, excluding satellites.

<void> walkTreeUnderConstruction(<*type(treeNode)> nextNode ↔) Return a pointer to the next node when performing a walk of a tree under construction.

<void> walkTreeWithSatellites(<*type(treeNode)> nextNode ↔) Return a pointer to the next node when performing a walk of the entire tree, including satellites.
```
16. Adding New Methods

16.1. 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:

```
!# <accretionDisksMethod>
!# <unitName>Accretion_Disks_Shakura_Sunyaev_Inititalize</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`.

16.2. 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;
16. Adding New Methods

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.

16.3. Components

This section describes the internal structure of node components, and how a component is implemented.

16.3.1. 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\(^1\), 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).

16.3.2. Extending Components

It is possible to create a component which extends an existing component (see the discussion of the extends element in §16.3.3). 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 (§11.9.2), which extends the scale dark matter profile component (§11.9.1). 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.

\(^1\)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.
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 §16.4.3), 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>
```
The elements of this document have the following meaning:
16.3. Components

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 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:

```plaintext
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.
16.3. 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 §16.4.3) 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 §16.3.3).
16. Adding New Methods

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:

```xml
<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 §16.3.3, 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:

```fortran
!# <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 §15.4 for a discussion of includeing C code within GALACTICUS.
16.3. 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 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

\[
D_i = \epsilon_{\text{abs}} s_i + \epsilon_{\text{rel}} |y_i|, \tag{16.1}
\]

where \( \epsilon_{\text{abs}} = [\text{odeToleranceAbsolute}] \), \( \epsilon_{\text{rel}} = [\text{odeToleranceRelative}] \), \( y_i \) 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 \texttt{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. \( y/y_i \) for the spheroid will be large). With just a relative tolerance (i.e. the \( \epsilon_{\text{rel}} |y_i| \) 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 \texttt{scale set} directive and associated subroutine should follow this template:

!# <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)


... 
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
```

16.4. Existing Method Types

16.4.1. Functions

Functions implement basic calculations (e.g. computing the power spectrum).

**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:

```fortran
!# <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:

```fortran
  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.
return function My_Accretion_Disk_Jet_Power_Get
end function My_Accretion_Disk_Jet_Power_Get

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:

\[
\dot{m}_{\text{minimum}} < \frac{M_\bullet}{M_{\text{Eddington}}} < \dot{m}_{\text{maximum}} \rightarrow \text{Shakura-Sunyaev} \\
\text{otherwise} \rightarrow \text{ADAF},
\] (16.2)

where $\dot{m}_{\text{minimum}} = \text{accretionRateThinDiskMinimum}$ and $\dot{m}_{\text{maximum}} = \text{accretionRateThinDiskMaximum}$ 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

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_Rate_Get
    Halo_Baryonic_Failed_Accreted_Mass_Get => My_Failed_Accreted_Mass_Get
    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
```

460
16.4. Existing Method Types

```fortran
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 and masses (assuring 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:

```fortran
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:

$$
\dot{M}_{\text{failed}} = \Omega_b \Omega_M \dot{M} - \dot{M}_{\text{accreted}},
$$

(16.3)

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:

```fortran
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
16. Adding New Methods

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:

**simple** 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 \( \dot{M}/M \). Abundances are computed assuming a pristine IGM (i.e. abundances are always zero) and chemicals are computed using the chemical state functions (see §16.4.1).

**null** Assumes no accretion onto halos.

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
```

462
16.4. Existing Method Types

Existing Method Types include:

- **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:

```plaintext
!# <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:

```plaintext
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:

```plaintext
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:

```
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:

```
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`. 
16.4. Existing Method Types

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:

```plaintext
!# <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
double precision function My_Bar_Instability_Timescale(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode

  return
end function My_Bar_Instability_Timescale
```

The function should compute and return 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.

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].
16. 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:

```plaintext
#!/ <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:

```plaintext
subroutine MethodInitialize(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 MethodInitialize
```

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:

```plaintext
double precision function My_Method_Get(thisNode, hostNode)
  implicit none
  type(treeNode), intent(inout), pointer :: 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 \[ \text{blackHoleInitialRadiusSpheroidRadiusRatio} \] 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`. 

466
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:

```plaintext
!# <blackHoleBinarySeparationGrowthRateMethod>
!#  <unitName>Black_Hole_Binary_Separation_Growth_Rate_Standard_Init</unitName>
!# </blackHoleBinarySeparationGrowthRateMethod>
```

Currently defined black hole binary separation growth rate methods are:

null Assumes 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:

```plaintext
!# <blackHoleBinaryRecoilVelocityMethod>
!#  <unitName>Black_Hole_Binary_Recoil_Velocity_Standard_Initialize</unitName>
!# </blackHoleBinaryRecoilVelocityMethod>
```

Currently defined black hole binary recoil velocity methods are:

null Assumes that there is zero recoil velocity.

Campanelli2008 Assumes 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:

```plaintext
!# <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:
16. Adding New Methods

```fortran
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:

```fortran
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:

```fortran
!# <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:

```fortran
  implicit none
  type(varying_string), intent(in) :: chemicalStateMethod
```

468
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 §16.5.2).

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$^{-3}$), composit-
tion as described by the **abundances** structure and in the presence of a radiation field described by the **radiation** structure.

Currently defined chemical state methods are:

**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:

```xml
<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>
  <description>Some description of what this chemical state is.</description>
  <extrapolation>
    <metallicity>
      <limit>low</limit>
      <method>fixed</method>
    </metallicity>
  </extrapolation>
  ...
</chemicalStates>
```
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\(_{\text{i}}\) and H\(_{\text{ii}}\) 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`, 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 software to compute the chemical state for atomic gas in collisional ionization equilibrium. CLOUDY will be downloaded, compiled and run automatically if necessary\(^5\).

- **PIE_from_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:

```hdf5
GROUP "/" {
    DATASET "neutralHydrogenRatio" {
        DATATYPE HST_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.
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);
16.4. Existing Method Types

- **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);

- **heliumToHydrogenRatio** 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 Stellar Mass Functions**

Additional methods for empirical conditional mass functions can be added using the `conditionalStellarMassFunctionMethod` 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
!# <conditionalStellarMassFunctionMethod>
!# <unitName>Conditional_Stellar_Mass_Functions_Behroozi2010_Initialize</unitName>
!# </conditionalStellarMassFunctionMethod>
```

Here, `Conditional_Stellar_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
implicit none

type(varying_string), intent(in) :: conditionalStellarMassFunctionMethod
procedure(double precision), pointer, intent(inout) :: Cumulative_Conditional_Stellar_Mass_Function_Get, Cumulative_Conditional_Stellar_Mass_Function_Var_Get

if (conditionalStellarMassFunctionMethod == 'myMethod') then
    Cumulative_Conditional_Stellar_Mass_Function_Get => My_Cumulative_Conditional_Stellar_Mass_Function
    Cumulative_Conditional_Stellar_Mass_Function_Var_Get => My_Cumulative_Conditional_Stellar_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 function must have the form:
16. Adding New Methods

```fortran
double precision function My_Cumulative_Conditional_Stellar_Mass_Function(massHalo,massStellar)
  implicit none
  double precision, intent(in) :: massHalo,massStellar
  
  return
end function My_Cumulative_Conditional_Stellar_Mass_Function

double precision My_Cumulative_Conditional_Stellar_Mass_Function_Var My_Cumulative_Conditional_Stellar_Mass_Function(massHalo,massStellarLow,massStellarHigh)
  implicit none
  double precision, intent(in) :: massHalo,massStellarLow,massStellarHigh
  
  return
end function My_Cumulative_Conditional_Stellar_Mass_Function_Var
```

The first function must return the number of galaxies of mass greater than \( \text{massStellar} \) in halos of mass \( \text{massHalo} \). The second function should return the variance in the number of galaxies in the stellar mass range \( \text{massStellarLow} \) to \( \text{massStellarHigh} \) in halos of mass \( \text{massHalo} \).

Currently defined tree timing methods are:

**Behroozi2010** This method uses the fitting function of Behroozi et al. [2010] to compute the conditional stellar 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:

```fortran
!# <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:

```fortran
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
16.4. Existing Method Types

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:

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 $M_\odot$/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

$$
\dot{M}_{cool} = 4\pi \rho(r_{\text{infall}})r_{\text{infall}}^2 (16.4)
$$

if the infall radius is within the outer radius of the hot halo and

$$
\dot{M}_{cool} = \frac{M_{\text{hot}}}{\tau_{\text{dynamical,halo}}} (16.5)
$$

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:

```
!# <coolingFunctionMethods>
!# <unitName>Cooling_Function_Atomic_CIE_CloudyInitializer</unitName>
!# </coolingFunctionMethods>
```

Here, `Cooling_Function_Atomic_CIE_CloudyInitializer` is the name of a subroutine which will be called to initialize the method. The initialization subroutine must have the following form:

```
subroutine MethodInitializer(coolingFunctionMethods,coolingFunctionsMatched)
  implicit none
  type(varying_string), intent(in ) :: coolingFunctionMethods(:)
  integer, intent(inout) :: coolingFunctionsMatched
```

475
16. Adding New Methods

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:

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\(^6\) 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.
16. Adding New Methods

Figure 16.1.: 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 necessary⁷. Figure 16.1 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].

⁷CLOUDY 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.
Cooling Radius

Additional methods for cooling radius calculations can be added using the \texttt{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 \texttt{simple} method is described by a directive:

\begin{verbatim}
!# <coolingRadiusMethod>
!# <unitName>Cooling_Radius_Simple.Initialize</unitName>
!# </coolingRadiusMethod>
\end{verbatim}

Here, \texttt{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:

\begin{verbatim}
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
  return
end subroutine Method.Initialize
\end{verbatim}

where \texttt{myMethod} is the name of this method as will be specified by the \texttt{coolingRadiusMethod} input parameter. The procedure pointer \texttt{Cooling_Radius_Get} must be set to point to a function which returns the cooling function as described below while \texttt{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:

\begin{verbatim}
double precision function Cooling_Radius_Get(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode

  return
end function Cooling_Radius_Get
\end{verbatim}

The function must return the cooling radius (in units of Mpc) for \texttt{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, $t_{\text{cool}} \propto \rho^{-1}$. Consequently, the cooling radius grows as the square root of the time available for cooling.
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:

```
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
  return
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:

```
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:
16.4. 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:

```fortran
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
```

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:

```fortran
double precision function Infall\_Radius\_Get(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode

  return
end function Infall\_Radius\_Get
```

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 Mpc/Gyr.

Currently defined infall radius methods are:

- \texttt{coolingRadius} Assumes that the infall radius equals the cooling radius.
- \texttt{cooling and freefall} Assumes that the infall radius is equal to the smaller of the cooling and freefall radii.

\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:
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>
```
16.4. Existing Method Types

Here, Cooling_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 \([\\text{coolingTimeAvailableAgeFactor}]\);

**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:

```fortran
!# <freefallTimeAvailableMethod>
!#   <unitName>Freefall_Time_Available_Halo_Formation_Initialize</unitName>
!# </freefallTimeAvailableMethod>
```
16. Adding New Methods

Here, \texttt{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:

\begin{verbatim}
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
\end{verbatim}

where \texttt{myMethod} is the name of this method as will be specified by the \texttt{freefallTimeAvailableMethod} input parameter. The procedure pointers \texttt{Freefall\_Time\_Available\_Get} and \texttt{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:

\begin{verbatim}
double precision function Freefall\_Time\_Available\_Get(thisNode)
    implicit none
    type(treeNode), intent(inout), pointer :: thisNode
    return
end function Freefall\_Time\_Available\_Get
\end{verbatim}

The first function must return the time available for freefall cooling calculations (in units of Gyr) for \texttt{thisNode}, while the second must return the rate of increase of this time.

Currently defined freefall time available methods are:

\texttt{haloFormation} The time available for cooling is set equal to the current time minus the formation time of the halo.

\section*{Cooling Time}

Additional methods for cooling time calculations can be added using the \texttt{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 \texttt{simple} method is described by a directive:

\begin{verbatim}
!# <coolingTimeMethod>
!# <unitName>Cooling\_Time\_Simple\_Initialize</unitName>
!# </coolingTimeMethod>
\end{verbatim}

Here, \texttt{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:
16.4. Existing Method Types

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(chemicalDensitiesStructure), 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:

#! <cosmologicalMassVarianceMethod>
#! <unitName>Cosmological_Mass_Variance_Filtered_Power_Spectrum.Initialize</unitName>
#! </cosmologicalMassVarianceMethod>

485
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 `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 §16.4.1).

**Cosmology**

Additional methods for cosmology can be added using the `cosmologyMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `matter-lambda` method is described by a directive:

```fortran
!# <cosmologyMethod>
!# <unitName>Cosmology_Functions_Matter_Lambda_Initialize</unitName>
!# </cosmologyMethod>
```
Here, \texttt{Cosmology\_Functions\_Matter\_Lambda\_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) :: cosmologyMethod
  procedure(), pointer, intent(inout) :: Early\_Time\_Density\_Scaling\_Get
if (cosmologyMethod == 'myMethod') then
  Procedure\_Pointer\_Get => My\_Procedure
end if
return
end subroutine Method\_Initialize
```

where \texttt{myMethod} is the name of this method as will be specified by the \texttt{cosmologyMethod} input parameter. Numerous procedure pointers are passed in and should be set to point to functions/subroutines that perform cosmological calculations as described below. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

Below are a list of procedure templates for the various procedures required by the cosmology method. After each template a description of what the procedure must do is given.

```fortran
logical function My\_Expansion\_Factor\_Is\_Valid(aExpansion)
  implicit none
  double precision, intent(in) :: aExpansion
  .
  .
  .
return
end function My\_Expansion\_Factor\_Is\_Valid
```

Should return true if the input expansion factor is valid (e.g. greater than zero and less than the maximum expansion factor for a collapsing universe).

```fortran
logical function My\_Cosmic\_Time\_Is\_Valid(time)
  implicit none
  double precision, intent(in) :: time
  .
  .
  .
```

487
16. Adding New Methods

function My_Cosmic_Time_Is_Valid

return
end function My_Cosmic_Time_Is_Valid

Should return true if the input cosmic time is valid (e.g., greater than zero and less than the time at the Big Crunch in collapsing universes).

function My_Cosmology_Age(aExpansion,collapsingPhase)

implicit none

double precision, intent(in) :: aExpansion
logical, intent(in), optional :: collapsingPhase

return
end function My_Cosmology_Age

Should return the age (in Gyr) of the universe at a given expansion factor. The collapsingPhase argument optionally specifies whether the age during the expanding or collapsing phase is required—if absent then the expanding phase should be assumed.

function My_Expansion_Factor(tCosmological)

implicit none

double precision, intent(in) :: tCosmological

return
end function My_Expansion_Factor

Should return the expansion factor of the universe at the input cosmological time (given in Gyr).

function My_Expansion_Rate(aExpansion)

implicit none

double precision, intent(in) :: aExpansion

return
end function My_Expansion_Rate

Should return the expansion rate, \( \dot{a}/a \) (in Gyr\(^{-1} \)), at the given expansion factor.

function My_Hubble_Parameter(tCosmological,aExpansion,collapsingPhase)

implicit none

double precision, intent(in), optional :: tCosmological,aExpansion
logical, intent(in), optional :: collapsingPhase

return
end function My_Hubble_Parameter
16.4. Existing Method Types

Should return the Hubble parameter (in km/s/Mpc) at the given cosmic epoch. The cosmic epoch can be specified as a time since the Big Bang, or an expansion factor (in the expanding phase unless the collapsing phase is specifically requested). Specifying a time and an expansion factor is invalid input, and the routine should abort in such cases.

```fortran
subroutine My_Early_Time_Density_Scaling(dominateFactor,densityPower,aDominant,Omega_Dominant)
    implicit none
    double precision, intent(in) :: dominateFactor
    double precision, intent(out) :: densityPower,aDominant
    double precision, intent(out), optional :: Omega_Dominant
    return
end subroutine My_Early_Time_Density_Scaling
```

Should compute the epoch, \(a\text{Dominant} \), before which the universe is dominated by a single component (e.g., in a matter and cosmological constant only universe, the epoch before which matter always dominated). The epoch should be such that the component in question dominates over others in density by a factor of \(\text{dominateFactor} \). The exponent in the density-expansion factor relation for this relation should be returned in \(\text{densityPower} \). If present, \(\text{Omega_Dominant} \) should be set to the density of the dominant component (in units of the critical density) at the present day.

```fortran
double precision function My_Epoch_of_Matter_Domination(dominateFactor)
    implicit none
    double precision, intent(in) :: dominateFactor
    return
end function My_Epoch_of_Matter_Domination
```

Should return the epoch at which matter dominates the energy density of the universe by the specified factor.

```fortran
double precision function My_Omega_Matter(tCosmological,aExpansion,collapsingPhase)
    implicit none
    double precision, intent(in), optional :: tCosmological,aExpansion
    logical, intent(in), optional :: collapsingPhase
    return
end function My_Omega_Matter
```

Should return \(\Omega_M \) at the given cosmic epoch. The cosmic epoch can be specified as a time since the Big Bang, or an expansion factor (in the expanding phase unless the collapsing phase is specifically requested). Specifying a time and an expansion factor is invalid input, and the routine should abort in such cases.

```fortran
double precision function My_Omega_Dark_Energy(tCosmological,aExpansion,collapsingPhase)
    implicit none
```

489
16. Adding New Methods

double precision, intent(in), optional :: tCosmological, aExpansion
logical, intent(in), optional :: collapsingPhase
.
.
return
end function My_Omega_Dark_Energy

Should return $\Omega_\Lambda$ at the given cosmic epoch. The cosmic epoch can be specified as a time since the Big Bang, or an expansion factor (in the expanding phase unless the collapsing phase is specifically requested). Specifying a time and an expansion factor is invalid input, and the routine should abort in such cases.

double precision function My_Epoch_of_Matter_Dark_Energy_Equality(requestType)
  implicit none
  integer, intent(in) :: requestType
  
  return
end function My_Epoch_of_Matter_Dark_Energy_Equality

Should return the cosmic epoch at which matter and dark energy have equal energy densities. If requestType is requestTypeTime (specified in the Cosmology_Functions_Parameters module) then cosmic time (in Gyr) should be returned, while if it is requestTypeExpansionFactor then expansion factor should be returned instead.

double precision function My_Epoch_of_Matter_Curvature_Equality(requestType)
  implicit none
  integer, intent(in) :: requestType
  
  return
end function My_Epoch_of_Matter_Curvature_Equality

Should return the cosmic epoch at which matter and curvature have equal energy densities. If requestType is requestTypeTime (specified in the Cosmology_Functions_Parameters module) then cosmic time (in Gyr) should be returned, while if it is requestTypeExpansionFactor then expansion factor should be returned instead.

double precision function My_CMB_Temperature(tCosmological, aExpansion, collapsingPhase)
  implicit none
  double precision, intent(in), optional :: tCosmological, aExpansion
  logical, intent(in), optional :: collapsingPhase
  
  return
end function My_CMB_Temperature

Should return the CMB temperature (in K) at the given cosmic epoch. The cosmic epoch can be specified as a time since the Big Bang, or an expansion factor (in the expanding phase unless the collapsing phase
is specifically requested). Specifying a time and an expansion factor is invalid input, and the routine should abort in such cases.

```fortran
double precision function My_Comoving_Distance(time)
    implicit none
    double precision, intent(in) :: time
    .
    .
    return
end function My_Comoving_Distance
```

Should return the comoving distance (in units of Mpc) to the given cosmic time (in units of Gyr).

```fortran
double precision function My_Time_From_Comoving_Distance(comovingDistance)
    implicit none
    double precision, intent(in) :: comovingDistance
    .
    .
    return
end function My_Time_From_Comoving_Distance
```

Should return the cosmic time (in units of Gyr) corresponding to the given comoving distance (in units of Mpc).

```fortran
double precision function My_Dark_Energy_Equation_Of_State(time,expansionFactor)
    implicit none
    double precision, intent(in ), optional :: time,expansionFactor
    .
    .
    return
end function My_Dark_Energy_Equation_Of_State
```

Should return the dark energy equation of state (i.e. \( \omega \) in \( P \propto \rho^\omega \)) at the specified \texttt{time/expansionFactor}. At least one of \texttt{time} and \texttt{expansionFactor} must be supplied.

```fortran
double precision function My_Dark_Energy_Exponent(time,expansionFactor)
    implicit none
    double precision, intent(in ), optional :: time,expansionFactor
    .
    .
    return
end function My_Dark_Energy_Exponent
```

Should return the exponent for the scaling of the dark energy density, \( f(a) \), such that \( \rho_{DE}(a) = \rho_{DE}(a = 1)a^f(a) \) at the specified \texttt{time/expansionFactor}. At least one of \texttt{time} and \texttt{expansionFactor} must be supplied.

Currently defined cosmology methods are:
Cosmological relations are computed assuming a universe that contains only matter and a cosmological constant.

Cosmological relations are computed assuming a universe that contains only matter and dark energy with an equation of state $w(a) = w_0 + w_1 a (1 - a)$.

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:

```
!# <criticalOverdensityMethod>
!# <unitName>Spherical_Collape_Delta_Critical_Initialize</unitName>
!# </criticalOverdensityMethod>
```

Here, `Spherical_Collape_Delta_Critical_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(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
    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 following form:

```fortran
subroutine Critical_Overdensity_Tabulate(time,deltaCritNumberPoints,deltaCritTime,deltaCritDeltaCrit)
    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 in `Method_Initialize` or other parts of the code).
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 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:

```
!# <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:

```fortran
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:

```fortran
double precision function Critical_Overdensity_Mass_Scaling_Get(mass)
implicit none
double precision, intent(in) :: mass
```
16. Adding New Methods

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:

```plaintext
!# <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:

```plaintext
implicit none
```

```plaintext
type(varying_string), intent(in) :: darkMatterProfileMethod
```

```plaintext
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
```

This subroutine should define the density profile method by setting the pointers for all required functions.

494
16.4. Existing Method Types

where \texttt{myMethod} is the name of this method as will be specified by the \texttt{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:

\begin{verbatim}
double precision function My_Dark_Matter_Profile_Property(thisNode, radius)
    implicit none
    type(treeNode), intent(inout), pointer :: thisNode
    double precision, intent(in) :: radius

    .

    return
end function My_Dark_Matter_Profile_Property
\end{verbatim}

These functions should compute and return the density (in units of $M_\odot \text{Mpc}^{-3}$), enclosed dark matter mass (in units of $M_\odot$), gravitational potential (in units of (km/s)$^2$) and circular velocity (in units of km/s) due to dark matter at the given radius (in units of Mpc) for \texttt{thisNode} respectively.

The freefall radius functions must have the form:

\begin{verbatim}
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
\end{verbatim}

These functions should compute and return the freefall radius (in units of Mpc), or its growth rate given a \texttt{time} available for freefall (in Gyr) for \texttt{thisNode} respectively.

The energy, energy growth rate and rotation velocity normalization functions must have the form:

\begin{verbatim}
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
\end{verbatim}
The energy functions should compute and return the energy (potential plus kinetic; in units of $M_\odot (\text{km/s})^2$) of the dark matter halo out to the virial radius of thisNode and the rate of change of that energy (in units of $M_\odot (\text{km/s})^2 \text{Gyr}^{-1}$) 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:

```fortran
!# <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:
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:

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 sing 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 Concentration

Additional methods for the dark matter density profile concentration can be added using the darkMatterConcentrationMethod 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:

#include <darkMatterConcentrationMethod>
#include <unitName>Dark_Matter_Concentrations_Gao2008_Initialize</unitName>
#include <darkMatterConcentrationMethod>
Here, Dark_Matter_Concentrations_Gao2008_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(darkMatterConcentrationMethod, Dark_Matter_Profile_Concentration_Get)
  implicit none
  type(varying_string), intent(in) :: darkMatterConcentrationMethod
  procedure(), pointer, intent(inout) :: Dark_Matter_Profile_Concentration_Get
  if (darkMatterConcentrationMethod == 'myMethod') then
    Dark_Matter_Profile_Concentration_Get => My_Concentration_Get
  end if
  return
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the darkMatterConcentrationMethod input parameter. The procedure pointer Dark_Matter_Profile_Concentration_Get must be set to point to a function which returns the concentration of a node. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The concentration function must have the form:

```fortran
double precision function My_Dark_Matter_Profile_Concentration(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  return
end function My_Dark_Matter_Profile_Concentration
```

The function should compute and return the concentration for thisNode.

Currently defined dark matter profile concentration methods are:

NFW1996 The concentration is computed using the algorithm given by Navarro et al. [1996];

Gao2008 The concentration is computed using a fitting function from Gao et al. [2008];

Zhao2009 The concentration is computed using a fitting function from Zhao et al. [2009]. Halo formation times (as defined by Zhao et al. 2009) are computed directly from the merger tree branch associated with each node. Where the tree does not exist or has insufficient resolution to contain the merging time the merging time will be found by extrapolation of the earliest resolved progenitor halo using the selected mass accretion history method (see §16.4.1);

Munoz-Cuartas2011 The concentration is computed using a fitting function from Muñoz-Cuartas et al. [2011];

Prada2011 The concentration is computed using a fitting function from Prada et al. [2011];
16.4. Existing Method Types

**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:

```plaintext
!# <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:

```plaintext
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 `myMethod` is the name of this method as will be specified by the `darkMatterShapeMethod` input parameter. The procedure pointer `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:

```plaintext
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
```

The function should compute and return the shape parameter for `thisNode`. Currently defined dark matter profile shape parameter methods are:

- **Gao2008** The shape parameter is computed using a fitting function from Gao et al. [2008] - see §12.7.4 for details.

**Dark Matter Halo Spin Distribution**

Additional methods for the dark matter density profile concentration 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 Gao2008 method is described by a directive:
16. Adding New Methods

Here, 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:

```fortran
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
```

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 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:

```fortran
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.
16.4. Existing Method Types

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:

```!
!# <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:

```fortran
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 \( M_\odot/\text{Gyr} \).

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:

```
Here, \texttt{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(excursionSetBarrierMethod\_Excursion\_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 \texttt{myMethod} is the name of this method as will be specified by the \texttt{excursionSetBarrierMethod} input parameter. The procedure pointers \texttt{Excursion\_Sets\_Barrier\_Get}, and \texttt{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 \texttt{barrierName} argument. 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:

```fortran
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 \texttt{variance} and \texttt{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.
16.4. Existing Method Types

**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:

```latex
!# <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
```

The function must return the first crossing probability rate per unit variance at the specified `variance`, `varianceProgenitor`, and `time`.
16. Adding New Methods

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** Assumes 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].

**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:

```fortran
!# <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:

```fortran
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
```

504
16.4. Existing Method Types

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:

```fortran
!# <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:

```fortran
!# <excursionSetBarrierRemapGradient>
!# <unitName>Method_Barrier_Gradient_Remap</unitName>
!# </excursionSetBarrierRemapGradient>
```
16. Adding New Methods

```
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:

```
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:

```
506
```
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 §16.4.3) 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 §16.4.3) 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 §16.4.3) 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 §16.4.3) 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 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:

```markdown
!# <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:

```fortran
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
```

---

507
16. Adding New Methods

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:

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:

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 Density Profile

Additional methods for the hot halo density profile can be added using the hotHaloDensityMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the coredIsothermal method is described by a directive:

# <hotHaloDensityMethod>
!# <unitName>Hot_Halo_Density_Cored_Isothermal</unitName>
!# </hotHaloDensityMethod>
Here, *Hot_Halo_Density_Cored_Isothermal* 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) :: hotHaloDensityMethod

procedure(), pointer, intent(inout) :: Hot_Halo_Density_Get, Hot_Halo_Density_Log_Slope_Get,
                                       Hot_Halo_Enclosed_Mass_Get,
                                       Hot_Halo_Profile_Rotation_Normalization_Get,
                                       Hot_Halo_Profile_Radial_Moment_Get

if (hotHaloDensityMethod == 'myMethod') then
    Hot_Halo_Density_Get => My_Method_Get
    Hot_Halo_Density_Log_Slope_Get => My_Method_Log_Slope_Get
    Hot_Halo_Enclosed_Mass_Get => My_Method_Enclosed_Mass_Get
    Hot_Halo_Profile_Rotation_Normalization_Get => My_Method_Rotation_Normalization_Get
    Hot_Halo_Profile_Radial_Moment_Get => My_Method_Radial_Moment_Get
end if
return
end subroutine Method_Initialize
```

where *myMethod* is the name of this method as will be specified by the *hotHaloDensityMethod* input parameter. The procedure pointer *Hot_Halo_Density_Get* must be set to point to a function which returns the hot halo density as described below while *Hot_Halo_Density_Log_Slope_Get* should be set to point to a function which returns the logarithmic slope of that profile. Similarly, *Hot_Halo_Enclosed_Mass_Get*, *Hot_Halo_Profile_Rotation_Normalization_Get*, and *Hot_Halo_Profile_Radial_Moment_Get* should be set to point to functions which return the mass enclosed within a specified radius, the normalization of the rotation speed for the halo (see below), and the radial moments of the density profile respectively. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The density function must have the form:

```fortran
double precision function Hot_Halo_Density_Get(thisNode, radius)
implicit none

type(treeNode), intent(inout), pointer :: thisNode

double precision, intent(in) :: radius

return
end function Hot_Halo_Density_Get
```

The function must return the density (in units of \(M_\odot \text{Mpc}^{-3}\)) of the hot halo at the specified radius (given in Mpc) for *thisNode*. The logarithmic slope function should have the same template but return the logarithmic slope of the density profile at the specified radius. The enclosed mass function must return the mass (in units of \(M_\odot\)) of the hot halo within the specified radius (given in Mpc).

The rotation velocity normalization function must have the form:

```fortran
double precision function My_Rotation_Normalization(thisNode)
implicit none

type(treeNode), intent(inout), pointer :: thisNode

return
```

The function must return the rotation velocity normalization (in units of \(\text{km s}^{-1}\)) at the specified radius (given in Mpc) for *thisNode*.
16. Adding New Methods

end function My_Rotation_Normalization

and should compute and return the normalization between rotation speed and mean specific angular
momentum (in units of Mpc$^{-1}$) of thisNode assuming that the hot gas profile rotates at the same
velocity at all radii.

The radial moment function must have the form:

double precision function My_Radial_Moment(thisNode,moment,radius)
  implicit none
  type (treeNode), intent(inout), pointer :: thisNode
  double precision , intent(in ) :: moment,radius
  return
end function My_Radial_Moment

and should compute and return the radial moment of the profile

$$\int_0^r \rho(r')r'^n dr',$$

where $n =$moment and $r =$radius.

Currently defined hot halo density profile methods are:

coredIsothermal Implements a cored isothermal profile for the hot halo. The density is given by

$$\rho(r) = \frac{\rho_0}{1 + (r/r_{\text{core}})^2},$$

where the normalization is chosen to ensure the correct hot halo mass within the outer radius of
the hot halo, and $r_{\text{core}}$ is set using the selected cored isothermal core radius method (see §12.16).

null A null implementation that assumes zero density always.

Hot Halo Density Profile: Cored Isothermal Core Radius

Additional methods for the core radius of cored isothermal hot halo density profiles can be added us-
ing the hotHaloCoredIsothermalCoreRadiiMethod directive. The directive should contain a single
argument, giving the name of a subroutine to be called to initialize the method. For example, the
virialRadiusFraction method is described by a directive:

```plaintext
!# <hotHaloCoredIsothermalCoreRadiiMethod>
!# <unitName>Hot_Halo_Density_Cored_Isothermal_Core_Radii_VF_Initialize</unitName>
!# </hotHaloCoredIsothermalCoreRadiiMethod>
```

Here, Hot_Halo_Density_Cored_Isothermal_Core_Radii_VF_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(hotHaloCoredIsothermalCoreRadiiMethod,Hot_Halo_Density_Cored_Isothermal_Core_Radius_Get)
  implicit none
  type(varying_string), intent(in) :: hotHaloCoredIsothermalCoreRadiiMethod
  procedure(), pointer, intent(inout) :: Hot_Halo_Density_Cored_Isothermal_Core_Radius_Get
  if (hotHaloCoredIsothermalCoreRadiiMethod == 'myMethod') Hot_Halo_Density_Cored_Isothermal_Core_Radius_Get => Hot_Halo_Density_Cored_Isothermal_Core_Radius_Virial_Fraction
  return
end subroutine Method_Initialize
```
16.4. Existing Method Types

where myMethod is the name of this method as will be specified by the hotHaloCoredIsothermalCoreRadiiMethod input parameter. The procedure pointer Hot_Halo_Density_Cored_Isothermal_Core_Radius_Get must be set to point to a function which returns the core radius of a cored isothermal hot gas density profile. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The core radius function must have the form:

```
double precision function Hot_Halo_Core_Radius_Get(thisNode)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  .
  .
  .
  return
end function Hot_Halo_Core_Radius_Get
```

The function must return the radius (in units of Mpc) of the cored isothermal hot halo for thisNode. Currently defined hot halo cored isothermal density profile core radii methods are:

- **virialRadiusFraction** Implements a core radius equal to a fraction \( \frac{\text{isothermalCoreRadiusOverVirialRadius}}{\text{node's virial radius}} \) of the node's virial radius.

- **growingCore** Implements a core radius equal to a fraction \( \frac{\text{isothermalCoreRadiusOverScaleRadius}}{\text{universal baryon fraction times their total mass}} \) 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 \( \frac{\text{isothermalCoreRadiusOverVirialRadiusMaximum}}{\text{node's virial radius}} \) times the node's virial radius.

### 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

511
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:

```fortran
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 $\frac{M_\odot \text{km}^2 \text{s}^{-1} \text{Mpc}^{-3}}{\text{cm}^2 \text{g}^{-1}}$).

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].

**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:

```fortran
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:

```fortran
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 Temperature Profile

Additional methods for the hot halo temperature profile can be added using the `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 **virial** method is described by a directive:

```
!# <hotHaloTemperatureMethod>
!# <unitName>Hot_Halo_Temperature_Virial</unitName>
!# </hotHaloTemperatureMethod>
```

Here, `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:

```fortran
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:

```fortran
double precision function Hot_Halo_Temperature_Get(thisNode, radius)
  implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  double precision, intent(in) :: radius
```


16. Adding New Methods

... 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:

```fortran
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
  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.

```fortran
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:

**Press-Schechter** Implements the bias resulting from the Press-Schechter [Press and Schechter, 1974] mass function [Mo and White, 1996].

**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:

```
!# <haloMassFunctionMethod>
!# <unitName>Halo_Mass_Function_Press_Schechter_Initialize</unitName>
!# </haloMassFunctionMethod>
```

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
```
16. Adding New Methods

return
end function Halo_Mass_Function_Differential_Get

The function should return the halo mass function, \( \frac{dn}{dM} \) (in units of \( \text{Mpc}^{-3} \text{M}_\odot^{-1} \)) at mass \( m \) and time \( t \).

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:

```plaintext
!# <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:

```plaintext
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:

```plaintext
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
```
16.4. Existing Method Types

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 \( \text{massMinimum} \) to \( \text{massMaximum} \).

Currently defined halo mass sampling density function methods are:

**powerLaw** The distribution of halo masses is such that the mass of the \( i \)th halo is

\[
M_{\text{halo},i} = \exp \left[ \ln(M_{\text{halo},\text{min}}) + \ln \left( M_{\text{halo,\text{max}}}/M_{\text{halo,\text{min}}} \right) x^i \right].
\]

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 §9.1.1 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
```
16. Adding New Methods

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 \( \text{lognormalSpinDistributionMedian} \) and dispersion in \( \ln \lambda \) of \( \text{lognormalSpinDistributionSigma} \), both of which are input parameters to GALACTICUS.

**Bett2007** Implements distribution from Bett et al. [2007] with parameter \( \lambda_0 = [\text{spinDistributionBett2007Lambda0}] \) and \( \alpha = [\text{spinDistributionBett2007Alpha}] \), both of which are input parameters to GALACTICUS.

### Halo Profiles

Additional methods for the halo profile can be added using the `haloProfileMethod` directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the `isothermal` method is described by a directive:

```plaintext
!# <haloProfileMethod>
!# <unitName>Dark_Matter_Profile_Isothermal_Initailize</unitName>
!# </haloProfileMethod>
```

Here, `Dark_Matter_Profile_Isothermal_Initailize` 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) :: darkMatterProfileMethod
  procedure(), pointer, intent(inout) :: ...

  if (darkMatterProfileMethod == 'myMethod') then
    Dark_Matter_Profile_Energy_Get => My_Energy_Procedure
    Dark_Matter_Profile_Energy_Growth_Rate_Get => My_Energy_Growth_Rate_Procedure
    Dark_Matter_Profile_Rotation_Normalization_Get => My_Rotation_Normalization_Procedure
    Dark_Matter_Profile_Radius_from_Specific_Angular_Momentum_Get => My_Radius_from_Specific_Angular_Momentum_Procedure
    Dark_Matter_Profile_Circular_Velocity_Get => My_Circular_Velocity_Procedure
    Dark_Matter_Profile_Potential_Get => My_Potential_Procedure
    Dark_Matter_Profile_Enclosed_Mass_Get => My_Enclosed_Mass_Procedure
  end if
  return
end subroutine Method_Initialize
```

where `myMethod` is the name of this method as will be specified by the `haloProfileMethod` input parameter. The procedure pointer `Dark_Matter_Profile_Energy_Get` must be set to point to a function which returns the energy of the halo in units of \( M_\odot \text{ km}^2 \text{ s}^{-1} \) while `Dark_Matter_Profile_Energy_Growth_Rate_Get` must point to a function which returns the rate of change of that energy. The `Dark_Matter_Profile_Rotation_Normalization_Get` should point to a function which provides the normalization of the rotation velocity vs. specific angular momentum relation for the halo such that, when multiplied by \( 4\pi r^2 \) \( d\rho(r)r \), it gives the angular momentum of material between \( r \) and \( r + dr \), i.e. it should return \( A \) such that:

\[
J = A(j) \int_0^{r_{vir}} 4\pi^2 d\rho(r)r, \tag{16.9}
\]

where we ignore any variation of angular momentum within angle in each spherical shell of matter (since it will cancel out in later calculations anyway). `Dark_Matter_Profile_Radius_from_Specific_Angular_Momentum_Get` must be set to point to a procedure which returns the radius in the halo at
which circular orbits have the specified specific angular momentum. `Dark_Matter_Profile_Circular_Velocity_Get` must be set to point to a procedure which returns the circular velocity in the halo at a given radius. `Dark_Matter_Profile_Potential_Get` must be set to point to a procedure which returns the gravitational potential in the halo at a given radius. `Dark_Matter_Profile_Enclosed_Mass_Get` must be set to point to a procedure which returns the mass enclosed in the halo at a given radius. The initialization subroutine should perform any other tasks required to initialize the module (such as reading parameters etc.).

The halo energy function must have the form:

```fortran
double precision function Dark_Matter_Profile_Energy_Get(thisNode)
    implicit none
    type(treeNode), intent(inout), pointer :: thisNode
    .
    .
    .

    return
end function Dark_Matter_Profile_Energy_Get
```

The function must return the energy of `thisNode` in units of $M_\odot \, \text{km}^2 \, \text{s}^{-1}$. The energy growth rate function should have the same template but return the rate of change of the energy in units of $M_\odot \, \text{km}^2 \, \text{s}^{-1} \, \text{Gyr}^{-1}$. The rotation normalization function has the same template but should return the normalization, $A$, in the relation $V_{\text{rot}} = Aj$ where $j$ is the mean specific angular momentum of the halo. $A$ should have units of Mpc$^{-1}$.

The halo circular velocity function must have the form:

```fortran
double precision function Dark_Matter_Profile_Circular_Velocity_Get(thisNode, radius)
    implicit none
    type(treeNode), intent(inout), pointer :: thisNode
    double precision, intent(in) :: radius
    .
    .
    .

    return
end function Dark_Matter_Profile_Circular_Velocity_Get
```

and should return the circular velocity (in km/s) at the specified `radius` (in Mpc) in `thisNode`. The halo enclosed mass function must have the form:

```fortran
double precision function Dark_Matter_Profile_Enclosed_Mass_Get(thisNode, radius)
    implicit none
    type(treeNode), intent(inout), pointer :: thisNode
    double precision, intent(in) :: radius
    .
    .
    .

    return
end function Dark_Matter_Profile_Enclosed_Mass_Get
```

and should return the enclosed mass (in $M_\odot$) at the specified `radius` (in Mpc) in `thisNode`. The halo potential function must have the form:

```fortran
double precision function Dark_Matter_Profile_Potential_Get(thisNode, radius)
    implicit none
    type(treeNode), intent(inout), pointer :: thisNode
    double precision, intent(in) :: radius
    .
    .
    .

    return
end function Dark_Matter_Profile_Potential_Get
```
16. Adding New Methods

```fortran
implicit none
type(treeNode), intent(inout), pointer :: thisNode
double precision, intent(in) :: radius
.
.
return
end function Dark_Matter_Profile_Potential_Get
```

and should return the gravitational potential (in \( \text{km}^2/\text{s}^2 \)) at the specified \( \text{radius (in Mpc)} \) in \text{thisNode}. The potential is conventionally chosen such that \( \Phi(\text{virial}) = -V_{\text{virial}}^2 \) so that the potential at infinity is zero if the halo profile is truncated at the virial radius. The “radius from specific angular momentum” function should have the form:

```fortran
double precision function Dark_Matter_Profile_Radius_from_Specific_Angular_Momentum_Get(thisNode,specificAngularMomentum)
implicit none
type(treeNode), intent(inout), pointer :: thisNode
double precision, intent(in) :: specificAngularMomentum
.
.
return
end function Dark_Matter_Profile_Radius_from_Specific_Angular_Momentum_Get
```

and should return the radius (in Mpc) at which a circular orbit in the halo has the specified \( \text{specificAngularMomentum (in units of km s}^{-1} \text{ Mpc)} \).

Currently defined halo profile methods are:

- **isothermal** Implements an isothermal \( (\rho \propto r^{-2}) \) halo density profile.

**Halo Virial Density Contrast**

Additional methods for the halo virial density contrast can be added using the \text{virialDensityContrastMethod} directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the \text{sphericalTopHat} method is described by a directive:

```fortran
!# <virialDensityContrastMethod>
!# <unitName>Spherical_Collape_Delta_Virial_Initialize</unitName>
!# </virialDensityContrastMethod>
```

Here, \text{Spherical_Collape_Delta_Virial_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(virialDensityContrastMethod,Virial_Density_Contrast_Tabulate)
implicit none
type(varying_string), intent(in) :: virialDensityContrastMethod
procedure(), pointer, intent(inout) :: Virial_Density_Contrast_Tabulate
.
.
if (virialDensityContrastMethod.eq.'myMethod') then
    Virial_Density_Contrast_Tabulate => My_Do_Tabulate
.
.
end subroutine Method_Initialize
```
where \texttt{myMethod} is the name of this method as will be specified by the \texttt{virialDensityContrastMethod} input parameter. The procedure pointer \texttt{Virial\_Density\_Contrast\_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 Virial\_Density\_Contrast\_Tabulate(time,deltaVirialTableNumberPoints,deltaVirialTime,deltaVirialDeltaVirial)
implicit none
    double precision, intent(in) :: time
    double precision, allocatable, dimension(:), intent(inout) :: deltaVirialTime,deltaVirialDeltaVirial
    integer, intent(out) :: deltaVirialTableNumberPoints
    .
    return
end subroutine Virial\_Density\_Contrast\_Tabulate
```

The subroutine must tabulate the virial overdensity in array \texttt{deltaVirialDeltaVirial()} as a function of wavenumber \texttt{deltaVirialTime()} (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 \texttt{deltaVirialNumberPoints}. The subroutine should ensure that the currently requested \texttt{time} is within the range of the tabulated function (preferably with some buffer).

Currently defined virial density contrast methods are:

- **sphericalTopHat**: The virial density contrast is computed for a Universe containing collisionless matter and a cosmological constant following the spherical top hat collapse model (see, for example, Percival 2005).

- **Bryan-Norman1998**: The virial density contrast is computed using the fitting functions of 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**: The virial density contrast is fixed at $[\text{virialDensityContrastFixed}]$, defined relative to \texttt{criticalDensity} and \texttt{meanDensity} as specified by $[\text{virialDensityContrastFixedType}]$.

- **Kitayama-Suto1996**: The virial density constrast is computed using the fitting formula of Kitayama and Suto [1996], and is therefore valid only for flat cosmological models.

### Initial Mass Function Functions

Each registered IMF must provide multiple functions, specified by the following directives:

```fortran
!# <imfRecycledInstantaneous>
!# <unitName>Star\_Formation\_IMF\_Recycled\_Instantaneous\_My\_IMF</unitName>
!# </imfRecycledInstantaneous>

!# <imfYieldInstantaneous>
!# <unitName>Star\_Formation\_IMF\_Yield\_Instantaneous\_My\_IMF</unitName>
!# </imfYieldInstantaneous>
```
16. 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
```

---

522
16.4. Existing Method Types

if (imfSelected == imfIndex) then
  .
  .
  imfMatched=.true.
end if
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) :: minimumMass

  if (imfSelected == imfIndex) then
    .
    .
    .
  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
  logical, intent(inout) :: imfMatched
  double precision, intent(in) :: imfMass
  double precision, intent(out) :: imfPhi

  if (imfSelected == imfIndex) then
    .
    .
    .
  end if
end subroutine Star_Formation_IMF_Phi_My_IMF
16. Adding New Methods

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_IMF_Yield_Instantaneous_My_IMF Return a suitable metal yield in yield for this IMF in the instantaneous recycling approximation.

Star_Formation_IMF_Recycled_Instantaneous_My_IMF Return a suitable recycled fraction in recycledFraction for this IMF in the instantaneous recycling approximation.

Star_Formation_IMF_Tabulate_My_IMF Allocate the imfMass() and 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_IMF_Minimum_Mass_My_IMF Return the lowest mass for which the IMF is non-zero.

Star_Formation_IMF_Maximum_Mass_My_IMF Return the largest mass for which the IMF is non-zero.

Star_Formation_IMF_Phi_My_IMF Return the IMF for the specified imfMass initial stellar mass.

Currently defined IMFs are described in §12.20.2.

Initial Mass Function Selection

Additional methods for selection of initial mass functions can be added using the 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 fixed method is described by a directive:

!# <imfSelectionMethod>
!#   <unitName>IMF_Select_Fixed_Initialize</unitName>
!# </imfSelectionMethod>

Here, 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:

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
16.4. Existing Method Types

where myMethod is the name of this method as will be specified by the imfSelectionMethod input parameter. The procedure pointer 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 imfNames() contains a list of all available IMF names and can be used for index determination.

The selection function must have the form:

```fortran
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.

### Intergalactic Medium State

Additional methods for the intergalactic medium state can be added using the intergalaticMediumStateMethod directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the RecFast method is described by a directive:

```fortran
!# <intergalaticMediumStateMethod>
!# <unitName>Intergalactic_Medium_State_RecFast_Initialize</unitName>
!# </intergalaticMediumStateMethod>
```

Here, Intergalactic_Medium_State_RecFast_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(intergalaticMediumStateMethod, Intergalactic_Medium_Electron_Fraction_Get&, Intergalactic_Medium_Temperature_Get)
  implicit none
  type(varying_string),         intent(in) :: intergalaticMediumStateMethod
  procedure(),                  pointer, intent(inout) :: Intergalactic_Medium_Electron_Fraction_Get, Intergalactic_Medium_Temperature_Get
  if (intergalaticMediumStateMethod == ’myMethod’) then
    Intergalactic_Medium_Electron_Fraction_Get => My_Intergalactic_Medium_Electron_Fraction
    Intergalactic_Medium_Temperature_Get => My_Intergalactic_Medium_Temperature
  end if
  return
end subroutine Method_Initialize
```
where myMethod is the name of this method as will be specified by the intergalacticMediumStateMethod input parameter. The procedure pointer Intergalactic_Medium_Electron_Fraction_Get must be set to point to a function which returns the electron fraction as described below, while Intergalactic_Medium_Temperature_Get must be set to point to a function which returns the temperature of the intergalactic medium.

Both functions must have the form:

```fortran
double precision function Property_Get(time)
    implicit none
    double precision, intent(in) :: time
    ...
    return
end function Property_Get
```

The electron fraction function must return the electron number density in units of the hydrogen number density in the intergalactic medium, while the temperature function should return the temperature of the intergalactic medium (in Kelvin) at the specified time.

Currently defined intergalactic medium state methods are:

**file** Reads a tabulation of the intergalactic medium state from a file and interpolates in the table to give a result. The XML file containing the table should have the following form:

```xml
<igm>
  <electronFraction>
    <datum>1.1590278</datum>
    <datum>1.1590278</datum>
    ...
  </electronFraction>
  <matterTemperature>
    <datum>27230.271</datum>
    <datum>27203.016</datum>
    ...
  </matterTemperature>
  <redshift>
    <datum>9990.00</datum>
    <datum>9980.00</datum>
    ...
  </redshift>
</igm>
```

The electronFraction and matterTemperature elements should contain datum elements listing the relevant quantity for each redshift listed in the redshift element.
RecFast The state of the intergalactic medium is computed using the RecFast code Seager et al. [2000], Wong et al. [2008], which will be automatically downloaded, patched and built if necessary. The results from RecFast are stored in a file which is then processed as in the file method described above.

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:

```fortran
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
16. Adding New Methods

time \texttt{growthTableTime}() (second dimension) and wavenumber \texttt{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 \texttt{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, \texttt{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 $(9\Omega_M/4.0d0)^{1/3}(H_0t)^{2/3}$ during the matter dominated regime.

Currently defined linear growth factor methods are:

\texttt{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 \texttt{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 \texttt{modifiedPress-Schechter} method is described by a directive:

```plaintext
!# <treeBranchingMethod>
!# <unitName>Modified_Press_Schechter_Branching_Initialize</unitName>
!# </treeBranchingMethod>
```

Here, \texttt{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:

```plaintext
subroutine Method_Initialize(treeBranchingMethod,Tree_Branching_Probability,Tree_Subresolution_Fraction,Tree_Branch_Mass,Tree_Maximum_Step)
  type(varying_string), intent(in) :: treeBranchingMethod
  procedure(), pointer, intent(inout) :: Tree_Branching_Probability,Tree_Subresolution_Fraction,Tree_Branch_Mass,Tree_Maximum_Step
  if (treeBranchingMethod == 'myMethod') then
    Tree_Branching_Probability => My_Branching_Probability
    Tree_Subresolution_Fraction => My_Subresolution_Fraction
    Tree_Maximum_Step => My_Maximum_Step
    Tree_Branch_Mass => My_Branch_Mass
  .
  .
end if
return
end subroutine Method_Initialize
```

where \texttt{myMethod} is the name of this method as will be specified by the \texttt{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:

```plaintext
double precision function My_Branch_Mass(haloMass,deltaCritical,massResolution,probability)
  double precision, intent(in) :: haloMass,deltaCritical,massResolution,probability
```

528
16.4. Existing Method Types

Tree_Branching_Probability must point to a function which returns 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 $\text{massResolution}$. Tree_Subresolution_Fraction must point to a function which returns 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 $\delta_{\text{critical}}$, 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 $\text{haloMass}$ at time $\delta_{\text{critical}}$ should be allowed to take. 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. 16.2.

**generalizedPress-Schechter** Branching probabilities are computed using excursion set barrier first crossing rates (computed using the selected *excursionSetFirstCrossingMethod*; see §16.4.1), modified by the selected *treeBranchingModifierMethod* (see §16.4.1).
Figure 16.2.: Progenitor mass functions at redshifts $z = 0.5, 1, 2$ and 4 (bottom to top) for halos of mass $10^{12\pm 0.151}$, $10^{13.5\pm 0.151}$ and $10^{15\pm 0.151} h^{-1} M_\odot$ (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).
16.4. Existing Method Types

**Merger Tree Branching Modifiers**

Additional methods for merger tree branching probability modifiers can be added using the `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:

```
#! <treeBranchingModifierMethod>
#! <unitName>Merger_Tree_Branching_Modifiers_Null_Initialize</unitName>
#! </treeBranchingModifierMethod>
```

Here, `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 `myMethod` is the name of this method as will be specified by the `treeBranchingModifierMethod` input parameter. The procedure pointer `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
```

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:
16. Adding New Methods

```fortran
!# <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:


**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:

```fortran
!# <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:
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:

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 §16.4.1) 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 §16.4.1). See §12.23.4 for details.

stateRestore Intended primarily for debugging purposes, this method will restore a tree whose complete internal state was written to file. See §12.23.5 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 §12.23.3 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:

```plaintext
!# <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:

```plaintext
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:

```plaintext
double precision function Nonlinear_Power_Spectrum(wavenumber,time)
  implicit none
  double precision, intent(in  ) :: wavenumber,time
  .
  .
  return
end subroutine Nonlinear_Power_Spectrum
```

The function must return the non-linear matter power spectrum, $P_{nl}(k)$, 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.
16.4. Existing Method Types

**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:

```
!# <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:

```fortran
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:

```
!# <chemicalRatesCompute>
!#   <unitName>Chemical_Hydrogen_Rates_Compute</unitName>
!# </chemicalRatesCompute>
```

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
```

535
16. Adding New Methods

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:

!# <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:

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
    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
16.4. Existing Method Types

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
    Power_Spectrum_Window_Function_Wavenumber_Maximum_Get => My_Window_Function_Maximum_Wavelength
  end if

return
end subroutine Method_Initialize
```

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:

```fortran
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:

```fortran
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. \(10^{30}\)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:

```bash
!# <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:
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:

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:

**powerLaw** The power spectrum is assumed to be a power law, possibly with a running index. It is defined by

\[ P(k) \propto k^{n_s + \ln(k/k_{\text{ref}})[dn/d\ln k]}, \]

where the parameters are specified by input parameters \( n_s \equiv \text{powerSpectrumIndex} \), \( k_{\text{ref}} \equiv \text{powerSpectrumReferenceWavenumber} \) and \( dn/d\ln k \equiv \text{powerSpectrumRunning} \).

**Radiation Components**

Radiation components (i.e., types of radiation field that may be added to any radiation object; see §16.5.3) are defined using a combination of several directives: radiationLabel, radiationSet, radiationTemperature and radiationFlux. For example, the cosmic microwave background radiation component is defined by the following set of directives:
16. Adding New Methods

!# <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, radiationLabel, should contain a single element, 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 radiationTypeCMB will be available within GALACTICUS to select the cosmic microwave background component). The other directives must all specify the same label element and additional give, in a unitName element, the name of a function/subroutine to be called to perform the relevant calculation.

The radiationSet directive must specify a subroutine with the following template:

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
end subroutine Radiation_Set

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:

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

540
16.4. Existing Method Types

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:

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 \([T_{CMB}](1 + z)\).

IGB The intergalactic background light, set using the method selected by \([\text{radiationIntergalacticBackgroundMethod}]\); see §16.4.1.

Radiation Components: Intergalactic Background

Additional methods for the intergalactic background radiation component can be added using the \texttt{radiationIntergalacticBackgroundMethod} directive. The directive should contain a single argument, giving the name of a subroutine to be called.
16. Adding New Methods

to initialize the method. For example, the file method is described by a directive:

```plaintext
!# <radiationInter galacticBackgroundMethod>
!#  <unitName>Radiation_IGB_File_Initialize</unitName>
!# </radiationInter galacticBackgroundMethod>
```

Here, 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:

```plaintext
subroutine Method_Initialize(radiationInter galacticBackgroundMethod,Radiation_Set_Intergalactic_Background_Do,Radiation_Flux_Intergalactic_Background_Do)
implicit none
  type(varying_string), intent(in) :: radiationInter galacticBackgroundMethod
  procedure(), pointer, intent(inout) :: Radiation_Set_Intergalactic_Background_Do,Radiation_Flux_Intergalactic_Background_Do
  if (radiationInter galacticBackgroundMethod == 'myMethod') then
    Radiation_Set_Intergalactic_Background_Do => My_Method_Set
    Radiation_Flux_Intergalactic_Background_Do => My_Method_Flux
  end if
end subroutine Method_Initialize
```

where myMethod is the name of this method as will be specified by the radiationInter galacticBackgroundMethod input parameter. The procedure pointers Radiation_Set_Intergalactic_Background_Do and 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:

```plaintext
subroutine My_Method_Set(thisNode,radiationProperties)
implicit none
  type(treeNode), intent(inout), pointer :: thisNode
  double precision, intent(inout), allocatable, dimension(:) :: radiationProperties
end subroutine My_Method_Set
```

and should set the radiation component as described in §16.4.1. The flux subroutine must have the form:

```plaintext
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
end subroutine My_Method_Flux
```

and should increment radiationFlux as described in §16.4.1.

Currently defined intergalactic background radiation methods are:

file The intergalatic 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:
16.4. Existing Method Types

```
<specnem>
  <!-- URL http://adsabs.harvard.edu/abs/1996ApJ...461...20H -->
  <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>Angstroms</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>
</specnem>
```

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\(^{-1}\).

**Ram Pressure Mass Loss Rates in Disks**

Additional methods for computing ram pressure induced mass loss rates in disks can be added using the ramPressureStrippingMassLossRateDisksMethod 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:

```
!# <ramPressureStrippingMassLossRateDisksMethod>
!# <unitName>Ram_Pressure_Stripping_Mass_Loss_Rate_Disks_Simple_Init</unitName>
!# </ramPressureStrippingMassLossRateDisksMethod>
```

Here, 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:

```
subroutine Method_Initialize(ramPressureStrippingMassLossRateDisksMethod, Ram_Pressure_Stripping_Mass_Loss_Rate_Disk_Get)
  implicit none
```
16. Adding New Methods

```fortran
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
```

where **myMethod** is the name of this method as will be specified by the `ramPressureStrippingMassLossRateDisksMethod` input parameter. The procedure pointer `Ram_Pressure_Stripping_Mass_Loss_Rate_Disk_Get` 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:

```fortran
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
```

The function must return the mass loss rate induced by ram pressure forces (in units of $M_\odot$/Gyr) for the disk in `thisNode`.

Currently defined ram pressure mass loss rate methods are:

- **simple** The mass loss rate scales in proportion to the ratio of ram pressure and gravitational restoring forces;
- **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 `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 **simple** method is described by a directive:

```fortran
!# <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:

```fortran
subroutine Method_Initialize(satelliteMergingMassMovementsMethod,Satellite_Merging_Mass_Movement_Get)
  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
```

544
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:

```fortran
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
.
.
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.
- **Baugh2005** 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, if the baryonic mass of the satellite exceeds a fraction `burstMassRatio` of the baryonic mass of the host and the gas fraction in the host exceeds `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 `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 `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>
```
16. Adding New Methods

Here, 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:

```
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 myMethod is the name of this method as will be specified by the satelliteMergingRemnantSizeMethod input parameter. The procedure pointer 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:

```
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 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:

```
!# <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:

```fortran
  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:

```fortran
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
```

The subroutine must compute properties of the merger progenitor galaxies in `satelliteNode` and `hostNode`: `satelliteMass` and `hostMass` are the total masses of the two galaxies; `satelliteSpheroidMass` and `hostSpheroidMass` are the masses of each galaxy that will end up in the spheroid of the merger remnant; `hostSpheroidMassPreMerger` is the mass of the host spheroid prior to the merger; `satelliteRadius` and `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); `remnantSpheroidMass` is the mass of the spheroid in the remnant; `remnantSpheroidGasMass` is the mass of gas in the spheroid of the remnant; and `angularMomentumFactor` gives the pseudo-specific angular momentum of the remnant in units of \( (G M_{\text{remnant}, \text{spheroid}} r_{\text{remnant}, \text{spheroid}})^{1/2} \) where \( M_{\text{remnant}, \text{spheroid}} \) is the mass of the remnant spheroid and \( r_{\text{remnant}, \text{spheroid}} \) 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 Merging Timescales

Additional methods for the satellite merging timescales can be added using the satelliteMergingMethod
directive. The directive should contain a single argument, giving the name of a subroutine to be called
to initialize the method. For example, the Lacey-Cole method is described by a directive:

    !# <satelliteMergingMethod>
    !# <unitName>Satellite_Time_Until_Merging_Lacey_Cole_Initialize</unitName>
    !# </satelliteMergingMethod>

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:

    subroutine Method_Initialize(satelliteMergingMethod,Satellite_Time_Until_Merging)
        implicit none
        type(varying_string), intent(in) :: satelliteMergingMethod
        procedure(), pointer, intent(inout) :: Satellite_Time_Until_Merging

        if (satelliteMergingMethod == 'myMethod') Satellite_Time_Until_Merging => My_Method_Procedure
        return
    end subroutine Method_Initialize

where myMethod is the name of this method as will be specified by the satelliteMergingMethod input
parameter. The procedure pointer Satellite_Time_Until_Merging 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 merging timescale function must have the form:

    double precision function Satellite_Time_Until_Merging(thisNode,thisOrbit)
        implicit none
        type(treeNode), pointer, intent(inout) :: thisNode
        type(keplerOrbit), intent(inout) :: thisOrbit

        return
    end function Satellite_Time_Until_Merging

The function must return the merging timescale for thisNode orbitting with thisOrbit at virial radius
crossing in thisNode%parentNode in units of Gyr.

Currently defined satellite virial orbit methods are:

Lacey-Cole  Computes the merging timescale using the method of Lacey and Cole [1993].
Lacey-Cole+Tormen Computes the merging timescale using the method 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];
Jiang2008 Computes the merging timescale using the method of Jiang et al. [2008];
BoylanKolchin2008 Computes the merging timescale using the method of Boylan-Kolchin et al. [2008];

**preset** 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;

**Wetzel-White** Computes the merging timescale using the method of Wetzel and White [2010].

### 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:

```plaintext
#!/ <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:

```plaintext
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:

```plaintext
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
```

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 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:

```plaintext
#!/ <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_Spheroids_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

  .
  .
  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:
16.4. Existing Method Types

**fixed** The outflow rate is a fixed multiple of the the star formation rate.

**powerLaw** The outflow rate is given by

\[
M_{\text{outflow}} = \left( \frac{V_{\text{outflow}}}{V} \right)^{\alpha_{\text{outflow}}} \frac{\dot{E}}{E_{\text{canonical}}},
\]

(16.11)

where \( V_{\text{outflow}} = \text{[disk|spheroid]OutflowVelocity} \) (in km/s) and \( \alpha_{\text{outflow}} = \text{[disk|spheroid]OutflowVelocity} \) 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 \( 1M_\odot \) 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:

```plaintext
!# <starFormationExpulsiveFeedbackSpheroidsMethod>
!# <unitName>Star_Formation_Expulsive_Feedback_Spheroids_Power_Law_Initilaize</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:

```plaintext
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]Method` 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:

```plaintext
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
```

---

\(^8\)“Expulsive” feedback implies outflows in which gas is driven not only out of a galaxy but also out of its host dark matter halo.
16. Adding New Methods

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 Assumed a zero outflow rate.

superwind The outflow rate is given by

$$ M_{\text{outflow}} = \beta_{\text{superwind}} \frac{\dot{E}}{E_{\text{canonical}}} \left\{ \begin{array}{ll} (V_{\text{superwind}}/V)^2 & \text{if } V > V_{\text{superwind}} \\ 1 & \text{otherwise,} \end{array} \right. $$

where $V_{\text{superwind}} = [\text{disk|spheroid}]\text{SuperwindVelocity}$ (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 $1M_\odot$ 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
```
16.4. Existing Method Types

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 §12.30.1).

**extendedSchmidt** The rate is given by the extended Schmidt law (Shi et al. 2011; see §12.30.2).

**Blitz-Rosolowsky2006** The rate is given by the Blitz-Rosolowsky rule (Blitz and Rosolowsky 2006; see §12.30.3).

**KMT09** The rate is given by the Krumholz-McKee-Tumlinson (Krumholz et al. 2009; see §12.30.4);

### 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:

```fortran
!# <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:

```fortran
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:

```fortran
double precision function Star_Formation_Timescale_Get(thisNode)
  implicit none
  type(treeNode), intent(in) :: thisNode
```

```fortran
end function Star_Formation_Timescale_Get
```
16. Adding New Methods

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

\[ \tau_\star = \epsilon_\star^{-1} \tau_{\text{dynamical,disk|spheroid}} \left( \frac{V_{\text{disk|spheroid}}}{200\text{ km/s}} \right)^{\alpha_\star}, \quad (16.13) \]

where \( \epsilon_\star \) is a star formation efficiency and \( \alpha_\star \) controls the scaling with velocity. Note that \( \tau_{\text{dynamical,disk|spheroid}} = R_{\text{disk|spheroid}} / V_{\text{disk|spheroid}} \) 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 \( \tau_\star = M_{\text{cold}} / \int_0^\infty 2\pi r \Sigma_\star (r) dr \) where \( \Sigma_\star (r) \) is the surface density of star formation rate (see §12.30)

**Stellar Astrophysics**

Additional methods for stellar astrophysical properties can be added using the `stellar AstrophysicsMethod` 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
!# <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:

```fortran
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
end if
return
end subroutine Method_Initialize
```
where \texttt{myMethod} is the name of this method as will be specified by the \texttt{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:

\begin{verbatim}
double precision function Star_Property(initialMass,metallicity)
  implicit none
  double precision, intent(in) :: initialMass,metallicity
  .
  .
  return
end function Star_Property
\end{verbatim}

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 \texttt{initialMass} and \texttt{metallicity}.

The metal yield function must have the form:

\begin{verbatim}
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
\end{verbatim}

This function must return the yield (in $M_\odot$) of the element identified by \texttt{atomIndex} (as returned by the \texttt{Atom Lookup()} function from the \texttt{Atomic Data} module) if present, or total metal yield otherwise for a star of the specified \texttt{initialMass} and \texttt{metallicity}.

The initial mass function must have the form:

\begin{verbatim}
double precision function Star_Initial_Mass(lifetime,metallicity)
  implicit none
  double precision, intent(in) :: lifetime,metallicity
  .
  .
  return
end function Star_Initial_Mass
\end{verbatim}

and should return the initial mass (in $M_\odot$) of a star of given \texttt{lifetime} (specified in Gyr) and \texttt{metallicity}.

Currently defined stellar astrophysics methods are:

\texttt{file} Stellar properties are read from an XML file and interpolated. The structure of the XML file is described in §12.35.1.

\textbf{Stellar Population Properties}

Additional methods for computing properties of stellar populations can be added using the \texttt{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 \texttt{instantaneous} method is described by a directive:
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:

```fortran
  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:

```fortran
subroutine Stellar_Population_Properties_Rates(starFormationRate, fuelAbundances, component, thisNode, thisHistory, stellarMassRate &, stellarAbundancesRates, stellarLuminositiesRates, fuelMassRate, fuelAbundancesRates, energyInputRate)
  implicit none
  double precision, intent(out) :: stellarMassRate, fuelMassRate, energyInputRate
  type(abundancesStructure), intent(out) :: stellarAbundancesRates, fuelAbundancesRates
  integer, intent(in) :: component
  double precision, intent(out), dimension(:) :: stellarLuminositiesRates
  double precision, intent(in) :: starFormationRate
  type(abundancesStructure), intent(in) :: fuelAbundances
  type(treeNode), intent(inout), pointer :: thisNode
  type(history), intent(inout) :: thisHistory
  .
  .
  return
end subroutine Stellar_Population_Properties_Rates
```
The subroutine is given the `starFormationRate` (in $M_\odot \, \text{Gyr}^{-1}$) 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 $M_\odot \, \text{Gyr}^{-1}$) in `stellarMassRate` and `fuelMassRate` respectively, and the corresponding rates (also in $M_\odot \, \text{Gyr}^{-1}$) of abundance change in `stellarAbundancesRates` and `fuelAbundancesRates` respectively. Finally, it should return rates of change (in $L_{AB} \, \text{Gyr}^{-1}$) 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:

```fortran
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 §16.3.3) 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

```fortran
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

```fortran
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 recycling approximation.
16. Adding New Methods

**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:

```plaintext
subroutine Stellar_Population_Spectrum_Tabulation(imfIndex, agesCount, metallicitiesCount, age, metallicity)
  implicit none
  integer, intent(in) :: imfIndex
```

558
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.

 fileSize 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 $L_\odot$ 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:

- **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;
- **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;}
16. Adding New Methods


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 Population Spectra Postprocessing

Additional methods for postprocessing spectra of stellar populations can be added using the stellarPopulationSpectraPostprocess directive. The directive should contain a single argument, giving the name of a subroutine to be called to initialize the method. For example, the Meiksin2006 method is described by a directive:

```
!# <stellarPopulationSpectraPostprocessInitialize>
!# <unitName>Stellar_Population_Spectra_Postprocess_Meiksin2006.Initialize</unitName>
!# </stellarPopulationSpectraPostprocessInitialize>
```

Here, Stellar_Population_Spectra_Postprocess_Meiksin2006.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(stellarPopulationSpectraPostprocessMethod, postprocessingFunction)
  implicit none
  type (varying_string), intent(in ) :: stellarPopulationSpectraPostprocessMethods
  procedure( ), pointer, intent(inout) :: postprocessingFunction

  if (stellarPopulationSpectraPostprocessMethod == 'myMethod') postprocessingFunction => myPostprocessor
  return
end subroutine Method_Initialize
```
where `myMethod` is the name of this method as will be specified by the `stellarPopulationSpectraPostprocessMethod` input parameter. The procedure pointer `postprocessingFunction` should be set to point to function with the following form:

```fortran
subroutine Method_Compute(wavelength,age,redshift,modifier)
    implicit none
    double precision, intent(in   ) :: wavelength,age,redshift
    double precision, intent(inout) :: modifier

    return
end subroutine Method_Compute
```

This function must multiply `modifier` by the factor by which stellar spectra will be scaled. The function is given the rest-frame `wavelength` (in Å), `age` of the population (in Gyr), and the `redshift` of the radiation and should return a multiplicative factor by which the spectrum will be scaled.

Currently defined stellar population postprocessing methods are:

- **Meiksin2006** Postprocesses spectra through absorption by the IGM using the results of Meiksin [2006].
- **Madau1995** Postprocesses spectra through absorption by the IGM using the results of Madau [1995].
- **lymanContinuumSuppress** Suppresses all luminosity in the Lyman continuum.
- **recent** Suppresses all from populations with ages in excess of `[recentPopulationsTimeLimit]`.
- **identity** This is an identity operator, i.e. it leaves the spectrum unchanged.

**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:

```fortran
!# <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:

```fortran
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
    end if
    return
end subroutine Method_Initialize
```
16. Adding New Methods

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:

```fortran
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:

```fortran
!# <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:

```fortran
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
    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:

```fortran
double precision function Stellar_Tracks_Function(initialMass,age,metallicity)
  implicit none
  double precision, intent(in) :: 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 §12.35.3.

### 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:

```fortran
!# <stellarWindsMethod>
!# <unitName>Stellar_Winds_Leitherer1992_Initialize</unitName>
!# </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:

```fortran
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
```
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:

```fortran
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 \(M_{\odot}/\text{Gyr}\)) 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_NagashimaInitialize</unitName>
!# </supernovaeIaMethod>
```

Here, **Supernovae_Type_Ia_NagashimaInitialize** 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(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
    .
    .
  end if
  return
end subroutine Method_Initialize
```
16.4. Existing Method Types

where \texttt{myMethod} is the name of this method as will be specified by the \texttt{supernovaeIaMethod} input parameter. The procedure pointers \texttt{SNeIa_Cumulative_Number_Get} and \texttt{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:

\begin{verbatim}
double precision function SNeIa_Cumulative_Number(initialMass,age,metallicity)
  implicit none
  double precision, intent(in) :: initialMass, age, metallicity
  .
  .
  return
end function SNeIa_Cumulative_Number
\end{verbatim}

and must return the number of Type Ia supernovae resulting per \( M_\odot \) of stars formed with given initial\texttt{Mass} and metallicity after a time \texttt{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:

\begin{verbatim}
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
\end{verbatim}

and should return the yield of the element identified by \texttt{atomIndex} (as returned by the \texttt{Atom_Lookup()} function from the \texttt{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:

\textbf{Nagashima} Computes Type Ia properties using the methods described by Nagashima et al. [2005].

\textbf{Tree Timing} Additional methods for tree timing (i.e. the time taken to process a given merger tree) can be added using the \texttt{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 \texttt{file} method is described by a directive:

\begin{verbatim}
!# <timePerTreeMethod>
!#  <unitName>Galacticus_Time_Per_Tree_File_Initialize</unitName>
!# </timePerTreeMethod>
\end{verbatim}

Here, \texttt{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:
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
    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 §17.2).

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
if (transferFunctionMethod == 'myMethod') then
    Transfer_Function_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 transferFunctionMethod input parameter. The procedure pointer Transfer_Function_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).

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.

CMBFast The transfer function is generated by CMBFAST 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:
16. Adding New Methods

<data>
  <column>\(k\) [Mpc^{-1}] - wavenumber</column>
  <column>T(\(k\)) - transfer function</column>
  <datum>1.11614e-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 CMBFast.</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>
  <wavenumber>
    <limit>high</limit>
    <method>power law</method>
  </wavenumber>
</extrapolation>
</data>
The **datum** elements give wavenumber (in $\text{Mpc}^{-1}$) 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.

### 16.4.2. 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:

```
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:

```
subroutine Events_Node_Merger_Do(thisNode)
    implicit none
    type(treeNode), intent(inout), pointer :: thisNode
    
    return
end subroutine Events_Node_Merger_Do_SLH
```
16. Adding New Methods

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.

#### 16.4.3. 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:

```plaintext
!# <calculationResetTask>
!#   <unitName>Tree_Node_Hot_Halo_Reset_Standard</unitName>
!# </calculationResetTask>
```

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:
16.4. Existing Method Types

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 \texttt{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:

\begin{verbatim}
!# <coolingRateModifierMethod>
!# <unitName>Cooling_Rate_Modifier_Cut_Off</unitName>
!# </coolingRateModifierMethod>
\end{verbatim}

Here, \texttt{Cooling_Rate_Modifier_Cut_Off} is the name of the subroutine which will be called to modify the cooling rate. The modification subroutine must have the following form:

\begin{verbatim}
subroutine Modify_Rate(thisNode,coolingRate)
    implicit none
    type(treeNode) , intent(inout), pointer :: thisNode
    double precision, intent(inout) :: coolingRate

    return
end subroutine Modify_Rate
\end{verbatim}

The subroutine should modify the \texttt{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 \texttt{[coolingCutOffVelocity]} at redshifts below/above \texttt{[coolingCutOffRedshift]} for \texttt{[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 \texttt{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:

\begin{verbatim}
!# <decodePropertyIdentifiersTask>
!# <unitName>Hernquist_Spheroid_Property_Identifiers_Decode</unitName>
!# </decodePropertyIdentifiersTask>
\end{verbatim}

Here, \texttt{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:
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

    !# <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:

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

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:

```fortran
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:

```fortran
!# <densityTask>
!# <unitName>Density_Procedure</unitName>
!# </densityTask>
```

where `Density_Procedure` is the name of a function with the following template

```fortran
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
```

If `componentType` is a match to the component then the function should return the density of the
component matching type `massType` at position for `thisNode`. `componentType` and `massType` can take
one of the values described in §16.2. In the above “density” can actually refer to different quantities
depending on the values of `weightBy` (and `weightIndex`):
16. Adding New Methods

**weightByMass** The actual mass should be returned (the value of weightIndex is irrelevant);

**weightByLuminosity** The weightIndex\textsuperscript{th} luminosity should be returned.

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 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:

```
!# <enclosedMassTask>
!# <unitName>Enclosed\_Mass\_Procedure</unitName>
!# </enclosedMassTask>
```

where \texttt{Enclosed\_Mass\_Procedure} is the name of a function with the following template

```
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
```

If componentType is a match to the component then the function should return the “mass” of the component matching type massType within radius for thisNode. componentType and massType can take one of the values described in §16.2. If radius is equal to or greater than radiusLarge the routine should the total “mass” (i.e. “mass” within infinite radius). In the above “mass” can actually refer to different quantities depending on the values of weightBy (and weightIndex):

**weightByMass** The actual mass should be returned (the value of weightIndex is irrelevant);

**weightByLuminosity** The weightIndex\textsuperscript{th} luminosity should be returned.

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 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:

```
!# <rotationCurveTask>
!# <unitName>Rotation\_Curve\_Procedure</unitName>
!# </rotationCurveTask>
```

where \texttt{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. componentType and massType can take one of the values described in §16.2. 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 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:

```plaintext
!# <rotationCurveGradientTask>
!# <unitName>Rotation_Curve_Gradient_Procedure</unitName>
!# </rotationCurveGradientTask>
```

where Rotation_Curve_Gradient_Procedure is the name of a function with the following template

```plaintext
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 §16.2. 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:
16. Adding New Methods

!# <potentialTask>
!# <unitName>Potential_Task</unitName>
!# </potentialTask>

where Potential_Task is the name of a function with the following template

```
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 §16.2. 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:

!# <surfaceDensityTask>
!# <unitName>Surface_Density_Procedure</unitName>
!# </surfaceDensityTask>

where Surface_Density_Procedure is the name of a function with the following template

```
double precision function Surface_Density_Procedure(thisNode, position, coordinateSystem, componentType, massType, haloLoaded)
  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
```

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 §16.2. 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_Formaltion_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>
```

The `<unitName>` tags specify subroutines that are called to register the IMF. These subroutines should have the following forms:
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-specifiable parameters then those parameters should be encoded into the descriptor.

Each registered IMF should supply a set of functions as described in §16.4.1.

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: \texttt{mergerTreeExtraOutputTask}. The directive should give the name of the subroutine to be called to perform the task. A template for this task is:

\begin{verbatim}
!# <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
\end{verbatim}

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 §16.4.3). If it is false then typically no output should occur (although other tasks may still be undertaken).
16.4. Existing Method Types

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:

```fortran
!# <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
type(treeNode), intent(inout), pointer :: thisNode
integer, intent(inout) :: integerProperty, integerBufferCount, doublePropertyCount
integer(kind=kind_int8), intent(inout) :: integerBuffer(:, :)
double precision, intent(inout) :: doubleBuffer(:, :)

return
end subroutine Galacticus_Output_Tree_Example
```
16. Adding New Methods

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:

```plaintext
!# <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:

```plaintext
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:

```plaintext
!# <mergerTreePostEvolveTask>
!# <unitName>Meta_Tree_Timing_Post_Evolve</unitName>
!# </mergerTreePostEvolveTask>
```

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:

---

\(^9\)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.
16.4. Existing Method Types

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:

```plaintext
!# <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:

```plaintext
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:

```plaintext
!# <satelliteMergerTask>
!# <unitName>Halo_Mass_Accretion_Rate</unitName>
!# </satelliteMergerTask>
```

Here, `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:

```plaintext
subroutine Merger_Tree_Initialize_Task(thisNode)
implicit none
type(treeNode), pointer, intent(inout) :: thisNode
.
.
return
end subroutine Merger_Tree_Initialize_Task
```
16. Adding New Methods

```
return
dend subroutine Merger_Tree_Initialize_Task
```

where `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 `mergerTreeStructureOutputTask`. The directive should give the name of the subroutine to be called to perform the task. The templates for this tasks is:

```
!# <mergerTreeStructureOutputTask>
!# <unitName>Structure_Output_Task</unitName>
!# </mergerTreeStructureOutputTask>
```

```
subroutine Structure_Output_Task(baseNode,nodeProperty,treeGroup)  
  implicit none  
  type(treeNode), intent(in), pointer :: baseNode  
  double precision, intent(inout), dimension(:) :: nodeProperty  
  type(hdf5Object), intent(inout) :: treeGroup  
  .  
  .  
  return  
end subroutine Structure_Output_Task
```

The subroutine must walk the merger tree beginning from the given `baseNode` and store each property to output in the given `nodeProperty` array. Once populated, this array can be written to the appropriate HDF5 group, given by `treeGroup`, in the GALACTICUS output file.

**Node Dump**

The function `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:

```
!# <nodeDumpTask>
!# <unitName>Node_Dump_Procedure</unitName>
!# </nodeDumpTask>
```

where `Node_Dump_Procedure` is the name of a subroutine with the following template

```
subroutine Node_Dump_Procedure(thisNode)  
  type(treeNode), intent(inout), pointer :: thisNode  
  .  
  .  
  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.
16.4. Existing Method Types

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 initialize 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 \texttt{mergerTreeOutputFilters} input parameter, are:

- \texttt{lightcone} Restricts output to those galaxies which fall within a specified lightcone geometry. See §13.7 for further details;
- \texttt{stellarMass} Restricts output to those galaxies with a total stellar mass equal to or greater than \texttt{[stellarMassFilterThreshold]};
- \texttt{luminosity} Restricts output to those galaxies with a total absolute AB magnitude less than or equal to\textsuperscript{10} \texttt{[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.

\textsuperscript{10}That is, galaxies which are at least as luminous as the specified threshold.
16. Adding New Methods

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:

```fortran
!# <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:

```fortran
!# <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:

```fortran
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.

Radius Solver Tasks
Galactic radii solver functions (see §16.4.1) 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:

```fortran
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:

```fortran
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 nod 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.
16. Adding New Methods

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:

```
!# <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:

```fortran
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:

```
!# <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:

```fortran
subroutine Satellite_Merger_Task(thisNode)
  implicit none
  type(treeNode), pointer, intent(inout) :: thisNode
  .
  .
  return
end subroutine Satellite_Merger_Task
```

where `thisNode` is the node about to merge with `thisNode%parentNode`. 
16.4. Existing Method Types

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:

```plaintext
!# <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:

```plaintext
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
```

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:

```plaintext
subroutine My_Create(thisNode, thisHistory)
implicit none
```

and should return a history object in `thisHistory` suitable for holding a star formation history for `thisNode`.

The `Star_Formation_History_Scales_Do` subroutine must have the form:
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 thisHistory, using the provided information on stellarMass and stellarAbundances if required.

The Star_Formation_History_Record_Do subroutine must have the form:

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 thisHistory for thisNode given the current starFormationRate and star formation fuelAbundances. That is, the subroutine should adjust the rates in thisHistory appropriately.

The Star_Formation_History_Output_Do subroutine must have the form:

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, thisHistory, for thisNode to the output file. The output number and tree index are provided as iOutput and treeIndex for reference, and 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" {
        COMMENT "Star formation histories for each tree."
        DATASET "diskSFH<nodeID>" {
          COMMENT "Star formation history stellar masses of the disk"
          DATATYPE H5T_IEEE_F64LE
        }
      }
    }
  }
}
where \texttt{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**
16. Adding New Methods

dataset is added to the starFormationHistories output group to record the metallicity binning as follows:

```plaintext
DATASET "metallicities" {
    COMMENT "Metallicities at which star formation histories are tabulated"
    DATATYPE H5T_IEEE_F64LE
    DATASPACE SIMPLE { ( [starFormationHistoryMetallicityCount] ) / ( [starFormationHistoryMetallicityCount] ) }
}
```

16.5. Subsystems

This section describes some of the subsystems within GALACTICUS that support various physical entities or processes.

16.5.1. 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, velocityTengentail, energy, angularMomentum, eccentricity, semiMajorAxis, radiusPericenter, radiusApoaenter. 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 \(\frac{GM_{\text{host}}}{r}\) where \(r\) is the radius of the orbit.

16.5.2. Chemicals

The chemicals subsystem provides both 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 described 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:

```plaintext
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
Adding New Methods

```plaintext
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.

#### 16.5.3. 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:

```plaintext
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,

```plaintext
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:

```plaintext
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.
16.5.4. 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 §16.5.5) 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:

```python
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:

```python
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

  ```python
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, $\rho(r) = \rho_0/[1 + (r/r_{core})^2]^{3/2}$. Initialization is by
16. Adding New Methods

```fortran
call myBetaProfile%initialize(beta, coreRadius, densityNormalization, mass, &
   outerRadius, isDimensionless)
```

\( \beta \) and \( r_{\text{core}} \) must always be specified. The density normalization must be specified either by the \texttt{densityNormalization} argument, or by supplying both \texttt{mass} and \texttt{outerRadius}. The profile will be assumed to be dimensionful unless the \texttt{isDimensionless} argument specifies otherwise.

\texttt{hernquist} \quad The Hernquist [Hernquist, 1990] profile. Initialization is by

```fortran
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 \texttt{isDimensionless} argument specifies otherwise.

\texttt{sersic} \quad The Sérsic [Sérsic, 1963] profile. Initialization is by

```fortran
call mySersicProfile%initialize(index, halfMassRadius, densityNormalization, mass, &
   isDimensionless)
```

The Sérsic \texttt{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 \texttt{isDimensionless} argument specifies otherwise.

16.5.5. Coordinates

The \texttt{coordinate} class, provided by the \texttt{Coordinates} module provides an object describing a position in three-dimensional space. Each extension of this class (currently, \texttt{coordinateCartesian}, \texttt{coordinateCylindrical}, and \texttt{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 \texttt{class(coordinate)} object can therefore convert it to, for example, spherical coordinates simply using

```fortran
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 §15.5.2 for complete details).
17. Auxilliary Methods

17.1. 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 [conditionalStellarMassFunctionMethod] input parameter.


Currently the only option, and selected using [conditionalStellarMassFunctionMethod] = Behroozi2010, this method adopts the fitting function of Behroozi et al. [2010]:

$$\langle N_c(M_\star|M) \rangle \equiv \int_{M_\star}^{\infty} \phi_c(M'_\star) d \ln M'_\star = \frac{1}{2} \left[ 1 - \text{erf} \left( \frac{\log_{10} M_\star - \log_{10} f_{\text{SHMR}}(M)}{\sqrt{2} \sigma_{\log M_\star}} \right) \right] ,$$  \hspace{1cm} (17.1)

Here, the function $f_{\text{SHMR}}(M)$ is the solution of

$$\log_{10} M = \log_{10} M_1 + \beta \log_{10} \left( \frac{M_\star}{M_{\star,0}} \right) + \frac{(M_\star/M_{\star,0})^\delta}{1 + (M_\star/M_{\star,0})^{-\gamma}} - 1/2 .$$  \hspace{1cm} (17.2)

For satellites,

$$\langle N_s(M_\star|M) \rangle \equiv \int_{M_\star}^{\infty} \phi_s(M'_\star) d \ln M'_\star = \langle N_c(M_\star|M) \rangle \left( \frac{f_{\text{SHMR}}^{-1}(M_\star)}{M_{\text{sat}}} \right)^{\alpha_{\text{sat}}} \exp \left( - \frac{M_{\text{cut}}}{f_{\text{SHMR}}^{-1}(M_\star)} \right) ,$$  \hspace{1cm} (17.3)

where

$$\frac{M_{\text{sat}}}{10^{12}M_\odot} = B_{\text{sat}} \left( \frac{f_{\text{SHMR}}^{-1}(M_\star)}{10^{12}M_\odot} \right)^{\beta_{\text{sat}}} ,$$  \hspace{1cm} (17.4)

and

$$\frac{M_{\text{cut}}}{10^{12}M_\odot} = B_{\text{cut}} \left( \frac{f_{\text{SHMR}}^{-1}(M_\star)}{10^{12}M_\odot} \right)^{\beta_{\text{cut}}} .$$  \hspace{1cm} (17.5)

By default, parameter values are taken from the fit of Leauthaud et al. [2011], specifically their 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 17.1. This method assumes that $P_c(N|M_\star,M;\delta \ln M_\star)$ 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

$$\phi(M_\star;M) \delta \ln M_\star = \sum_{N=0}^{\infty} NP(N|M_\star,M;\delta \ln M_\star)$$  \hspace{1cm} (17.6)
Table 17.1.: 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>$\alpha_{\text{sat}}$</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>$\log_{10} M_{*,0}$</td>
<td>10.916</td>
<td>[conditionalStellarMassFunctionBehrooziLog10Mstar0]</td>
</tr>
<tr>
<td>$\beta$</td>
<td>0.457</td>
<td>[conditionalStellarMassFunctionBehrooziBeta]</td>
</tr>
<tr>
<td>$\delta$</td>
<td>0.5666</td>
<td>[conditionalStellarMassFunctionBehrooziDelta]</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>1.53</td>
<td>[conditionalStellarMassFunctionBehrooziGamma]</td>
</tr>
<tr>
<td>$\sigma_{\log M_*}$</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>[conditionalStellarMassFunctionBehrooziBetaSatellite]</td>
</tr>
</tbody>
</table>

17.2. 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.

17.2.1. 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

$$\log_{10}[\tau_{\text{tree}}(M)] = \sum_{i=0}^{2} C_i (\log_{10} M)^i,$$

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).
18. 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

18.1. Program units

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: 120
program: tests_excursion_sets
Description: Computes various quantities related to the excursion set formalism and stores them in an output file.
Code lines: 100
Contained by: file Excursion_Sets.F90
Modules used: cosmological_parameters cosmology_functions
excursion_sets_barriers excursion_sets_first_crossings
galacticus_display galacticus_error
halo_mass_function input_parameters
io_hdf5 iso_varying_string
memory_management numerical_ranges
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
input_parameters iso_varying_string
memory_management
18. Source Code Documentation

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: 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: 52

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: 31
Contained by: file Millennium_Merger_Tree_File_Maker.F90
Modules used: command_arguments memory_management
merger_tree_data_structure merger_trees_millennium

Description: Contains a program which computes the optimal sampling of the halo mass function to minimize errors on the stellar mass function.
Code lines: 135

program: optimal_sampling_smf
Description: Compute the optimal number of trees to run of each mass.
Code lines: 114
Contained by: file Optimal_Sampling.Stellar_Mass_Function.F90
Modules used: cosmology_functions fgs1
galacticus_meta_compute_times halo_mass_function
input_parameters iso_c_binding
iso_varying_string memory_management
merger_trees_mass_function_sampling 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_stellar_mass_functions

file: Power_Spectra.F90
18.1. Program units

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
Contained by: file Power_Spectra.F90
Modules used: galacticus_error input_parameters
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: 58

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: 38
Contained by: file Simple_Merger_Tree_File_Maker.F90
Modules used: command_arguments 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: 161

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
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: 79

Contained by: file XRay_Absorption_ISM_Wilms2000.F90
18.1. Program units

Modules used: atomic_cross_sections_compton dates_and_times
io_hdf5 numerical_constants_prefixes
numerical_constants_units numerical_ranges

subroutine: xwrite
Description: Message display function required by dotbabs.
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 calculations of baryonic accretion into halos.
Code lines: 244

module: accretion_halos
Description: Implements calculations of baryonic accretion into halos.
Code lines: 224
Contained by: file accretion.halo.F90
Modules used: abundances_structure chemical_abundances_structure
galacticus_nodes iso_varying_string
Used by: subroutine node_component_hot_halo-
standard_rate_compute subroutine node_component_hot_halo-
standard_tree_initialize

subroutine: accretion_halos_initialize
Description: Initialze the accretion disk module.
18. Source Code Documentation

**subroutine:** halo_baryonic_accreted_abundances  
*Description:* Compute the mass of abundances (in $M_\odot$) accreted onto thisNode.  
*Code lines:* 13  
*Contained by:* module accretion_halos

**subroutine:** halo_baryonic_accreted_chemicals  
*Description:* Compute the mass of chemicals (in $M_\odot$) accreted onto thisNode.  
*Code lines:* 13  
*Contained by:* module accretion_halos

**function:** halo_baryonic_accreted_mass  
*Description:* Computes the mass of baryons accreted (in $M_\odot$) into thisNode from the intergalactic medium. Used to initialize nodes.  
*Code lines:* 13  
*Contained by:* module accretion_halos

**function:** halo_baryonic_accretion_rate  
*Description:* Computes the rate of baryonic mass accretion (in $M_\odot$/Gyr) onto thisNode from the intergalactic medium.  
*Code lines:* 12  
*Contained by:* module accretion_halos

**subroutine:** halo_baryonic_accretion_rate_abundances  
*Description:* Compute the rate of mass accretion of abundances (in $M_\odot$/Gyr) accreted onto thisNode.  
*Code lines:* 13  
*Contained by:* module accretion_halos

**subroutine:** halo_baryonic_accretion_rate_chemicals  
*Description:* Compute the mass of chemicals (in $M_\odot$) accreted onto thisNode.  
*Code lines:* 13  
*Contained by:* module accretion_halos

**function:** halo_baryonic_failed_accreted_mass  
*Description:* Computes the mass of baryons that failed to accrete (in $M_\odot$) into thisNode from the intergalactic medium. Used to initialize nodes.  
*Code lines:* 13  
*Contained by:* module accretion_halos

**function:** halo_baryonic_failed_accretion_rate  
*Description:* Computes the rate of failed baryonic mass accretion (in $M_\odot$/Gyr) onto thisNode from the intergalactic medium.  
*Code lines:* 12  
*Contained by:* module accretion_halos

**file:** accretion.halo.null.F90
**18.1. Program units**

*Description:* Contains a module which implements calculations of null baryonic accretion.

*Code lines:* 146

**module: accretion_halos_null**

*Description:* Implements calculations of null baryonic accretion.

*Code lines:* 126

*Contained by:* file `accretion.halo.null.F90`

*Modules used:* `abundances_structure`

*Used by:* subroutine `accretion_halos_null_initialize`

**subroutine: accretion_halos_null_initialize**

*Description:* Test if this method is to be used and set procedure pointer appropriately.

*Code lines:* 26

*Contained by:* module `accretion_halos_null`

*Modules used:* `iso_varying_string`

**subroutine: halo_baryonic_accreted_abundances_null_get**

*Description:* Computes the mass of abundances accreted (in $M_\odot$) onto thisNode from the intergalactic medium.

*Code lines:* 9

*Contained by:* module `accretion_halos_null`

*Modules used:* `galacticus_nodes`

**subroutine: halo_baryonic_accreted_chemicals_null_get**

*Description:* Computes the mass of chemicals accreted (in $M_\odot$) onto thisNode from the intergalactic medium.

*Code lines:* 10

*Contained by:* module `accretion_halos_null`

*Modules used:* `chemical_abundances_structure galacticus_nodes`

**function: halo_baryonic_accreted_mass_null_get**

*Description:* Computes the mass of baryons accreted into thisNode.

*Code lines:* 8

*Contained by:* module `accretion_halos_null`

*Modules used:* `galacticus_nodes`

**subroutine: halo_baryonic_accretion_rate_abundances_null_get**

*Description:* Computes the rate of mass of abundance accretion (in $M_\odot$/Gyr) onto thisNode from the intergalactic medium.

*Code lines:* 9

*Contained by:* module `accretion_halos_null`

*Modules used:* `galacticus_nodes`

**subroutine: halo_baryonic_accretion_rate_chemicals_null_get**

*Description:* Computes the rate of mass of chemicals accretion (in $M_\odot$/Gyr) onto thisNode from the intergalactic medium.

*Code lines:* 10

*Contained by:* module `accretion_halos_null`

*Modules used:* `chemical_abundances_structure galacticus_nodes`
function: halo_baryonic_accretion_rate_null_get
Description: Computes the baryonic accretion rate onto thisNode.
Code lines: 8
Contained by: module accretion_halos_null
Modules used: galacticus_nodes

function: halo_baryonic_failed_accreted_mass_null_get
Description: Computes the mass of baryons accreted into thisNode.
Code lines: 8
Contained by: module accretion_halos_null
Modules used: galacticus_nodes

function: halo_baryonic_failed_accretion_rate_null_get
Description: Computes the baryonic accretion rate onto thisNode.
Code lines: 8
Contained by: module accretion_halos_null
Modules used: galacticus_nodes

file: accretion.halo.simple.F90
Description: Contains a module which implements calculations of baryonic accretion onto halos using a simple truncation to mimic reionization.
Code lines: 365

module: accretion_halos_simple
Description: Implements calculations of baryonic accretion onto halos using a simple truncation to mimic reionization.
Code lines: 344
Contained by: file accretion.halo.simple.F90
Modules used: abundances_structure radiation_structure
Used by: subroutine accretion_halos_initialize

subroutine: accretion_halos_simple_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 89
Contained by: module accretion_halos_simple
Modules used: atomic_data chemical_abundances_structure cosmology_functions galacticus_error input_parameters intergalactic_medium_state iso_varying_string

subroutine: get_chemical_masses
Description: Compute the masses of chemicals accreted (in $M_{\odot}$) onto thisNode from the intergalactic medium.
Code lines: 38
Contained by: module accretion_halos_simple
Modules used: chemical_abundances_structure chemical_reaction_rates_utilities chemical_states cosmological_parameters dark_matter_halo_scales galacticus_nodes
18.1. Program units

numerical_constants_astronomical

subroutine: halo_baryonic_accreted_abundances_simple_get
Description: Computes the mass of abundances accreted (in $M_\odot$) onto thisNode from the intergalactic medium.
Code lines: 10
Contained by: module accretion_halos_simple
Modules used: galacticus_nodes

subroutine: halo_baryonic_accreted_chemicals_simple_get
Description: Computes the mass of chemicals accreted (in $M_\odot$) onto thisNode from the intergalactic medium.
Code lines: 22
Contained by: module accretion_halos_simple
Modules used: chemical_abundances_structure galacticus_nodes

function: halo_baryonic_accreted_mass_simple_get
Description: Computes the mass of baryons accreted into thisNode.
Code lines: 20
Contained by: module accretion_halos_simple
Modules used: cosmological_parameters dark_matter_halo_scales galacticus_nodes

subroutine: halo_baryonic_accretion_rate_abundances_simple_get
Description: Computes the rate of mass of abundance accretion (in $M_\odot$/Gyr) onto thisNode from the intergalactic medium.
Code lines: 10
Contained by: module accretion_halos_simple
Modules used: galacticus_nodes

subroutine: halo_baryonic_accretion_rate_chemicals_simple_get
Description: Computes the rate of mass of chemicals accretion (in $M_\odot$/Gyr) onto thisNode 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: 24
Contained by: module accretion_halos_simple
Modules used: chemical_abundances_structure galacticus_nodes

function: halo_baryonic_accretion_rate_simple_get
Description: Computes the baryonic accretion rate onto thisNode.
Code lines: 33
Contained by: module accretion_halos_simple
Modules used: cosmological_parameters dark_matter_halo_scales galacticus_nodes

function: halo_baryonic_failed_accreted_mass_simple_get
Description: Computes the mass of baryons accreted into thisNode.
Code lines: 20
Contained by: module accretion_halos_simple
18. Source Code Documentation

**Modules used:**
- cosmological_parameters
- dark_matter_halo_scales
- galacticus_nodes

**function:** halo_baryonic_failed_accretion_rate_simple_get

*Description:* Computes the baryonic accretion rate onto thisNode.

*Code lines:* 30

*Contained by:* module accretion_halos_simple

**Modules used:**
- cosmological_parameters
- 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

*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.
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: Returns 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).
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.

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.

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.

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`.

function: `adaf_temperature`

function: `adaf_v`
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`.

function: `black_hole_spin_up_rate_adaf`
Description: Computes the spin up rate of the black hole in `thisBlackHole` due to accretion from an ADAF disk.

file: `accretion_disks.Eddington_limited.F90`
18.1. Program units

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

Contains 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

Contains 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

Contains by: module accretion_disks_eddington

Modules used: galacticus_nodes

subroutine: accretion_disks_eddington_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.

Code lines: 38

Contains by: module accretion_disks_eddington

Modules used: input_parameters iso_varying_string

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

Contains 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

Contains by: file accretion_disks.F90

Modules used: iso_varying_string
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
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_adaf subroutine accretion_disks_initialize
Modules used: 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
18.1. Program units

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.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.
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: 40
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: 34
Contained by: module accretion_disks_switched
Modules used: black_hole_fundamentals galacticus_nodes

function: black_hole_spin_up_rate_switched
Description: Computes the spin up rate of the black hole in this black hole 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
18.1. Program units

**Contained by:** file `atomic.cross_sections.ionization.photo.F90`

**Modules used:** `iso_varying_string`

**Used by:** function `cross_section_h_gamma_to_-hplus_electron`

**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

**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
18.1. Program units

**Description:** Initializes the “Verner” atomic photo-ionization cross section module.

**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:** subroutine `accretion_halos_simple_initialize`

**Used by:** subroutine `abundances_set_metallicity`

**Used by:** subroutine `stellar_astrophysics_file_initialize`

**Used by:** subroutine `supernovae_type_ia_nagashima_initialize`

**Used by:** 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`
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.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: 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_-_galacticus_error
vernerr
input_parameters

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: 496

module: atomic_rates_ionization_collisional_verner
18.1. Program units

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: 474

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.

Code lines: 34

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.radiative.F90

Description: Contains a module that implements calculations of atomic radiative recombination rates.

Code lines: 113

module: atomic_rates_recombination_radiative

Description: Implements calculations of atomic radiative recombination rates.

Code lines: 93

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: atomic_rate_recombination_radiative

Code lines: 12

Contained by: module atomic_rates_recombination_radiative

subroutine: atomic_rate_recombination_radiative_initialize

Description: Initialize the atomic radiative recombination rate module.

Code lines: 53

Contained by: module atomic_rates_recombination_radiative

Modules used: atomic_rates_recombination_radiative galacticus_error radiative_verner 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: 597
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: 576

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.

Code lines: 50

Contained by: module atomic_rates_recombination_radiative_verner

Modules used: 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

Modules used: iso_varying_string

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
18.1. Program units

**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

**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
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: 63

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: 41

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: 17

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
18.1. Program units

**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 an 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_tidal_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`

**file:** `black_holes.binaries.recoil_velocity.Campanelli2007.F90`

**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 \( H = (7.3 \pm 0.3) \times 10^3 \text{ km/s} \), Gonzalez et al. [2007b] for \( A = 1.2 \times 10^4 \text{ km/s} \) \( B = -0.93 \), Gonzalez et al. [2007a] for \( K \cos(\delta \theta) = (6, -5.3) \times 10^4 \text{ km/s} \) and \( K = (6.0 \pm 0.1) \times 10^4 \text{ km/s} \).
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
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
Modules used: pseudo_random

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_standard_merge_black_holes
          subroutine node_component_black_hole_standard_satellite_merging
          subroutine black_hole_binary_recoil_velocity
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
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_null
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
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

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_null

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: 197

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:** 173

**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: 114
Contained by: module black_hole_binary_separations_standard
Modules used:
dark_matter_halo_scales
galactic_structure_options
galactic_structure_rotation_curves
galactic_structure_rotation_curve_gradients
galactic_structure_rotation_curves

galactic_structure_velocity_gradients

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

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

subroutine node_component_black_hole_simple_satellite_merging
subroutine node_component_black_hole_standard_satellite_merging
subroutine node_component_black_hole_standard_merge_black_holes

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
18.1. Program units

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

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 accretion_disk_jet_power_eddington

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_simple_rotation_curve_gradient

function node_component_black_hole_standard_potential

function node_component_black_hole_standard_rotation_curve

function node_component_black_hole_standard_rotation_curve_gradient

function adaf_bh_jet_power

function adaf_enthalpy-angular_momentum_product

function adaf_fluid-angular_velocity

function adaf_height

function adaf_v

function accretion_disk_jet_power_shakura_sunyaev

function black_hole_spin_up_rate_shakura_sunyaev

function accretion_disk_switched_adaf_radiative_efficiency_scaling

function node_component_black_hole_simple_rotation_curve

function node_component_black_hole_standard_mass_accretion_rate

function node_component_black_hole_standard_rotation_curve

program test_black_hole_fundamentals
function: \( a_1 \)
Description: Return the function \( A_1(j) \) that appears in the Kerr metric with spin \( \text{blackHoleSpin} \).
Code lines: 7
Contained by: module \texttt{black\_hole\_fundamentals}

function: \( a_2 \)
Description: Return the function \( A_2(j) \) that appears in the Kerr metric with spin \( \text{blackHoleSpin} \).
Code lines: 7
Contained by: module \texttt{black\_hole\_fundamentals}

function: \texttt{black\_hole\_eddington\_accretion\_rate}  
Description: Return the Eddington accretion rate (in \( M_\odot \text{Gyr}^{-1} \)) for the black hole in \texttt{thisBlackHole}.
Code lines: 9
Contained by: module \texttt{black\_hole\_fundamentals}
Modules used: \texttt{numerical\_constants\_astronomical} \texttt{numerical\_constants\_physical}

interface: \texttt{black\_hole\_frame\_dragging\_frequency}  
Code lines: 3
Contained by: module \texttt{black\_hole\_fundamentals}

function: \texttt{black\_hole\_frame\_dragging\_frequency\_node}  
Description: Returns the frame-dragging angular velocity in the Kerr metric.
Code lines: 32
Contained by: module \texttt{black\_hole\_fundamentals}
Modules used: \texttt{galacticus\_error}

function: \texttt{black\_hole\_frame\_dragging\_frequency\_spin}  
Description: Returns the frame-dragging angular velocity in the Kerr metric.
Code lines: 7
Contained by: module \texttt{black\_hole\_fundamentals}

function: \texttt{black\_hole\_gravitational\_radius}  
Description: Computes the gravitational radius (in Mpc) for the \texttt{thisBlackHole}.
Code lines: 8
Contained by: module \texttt{black\_hole\_fundamentals}
Modules used: \texttt{numerical\_constants\_physical}

interface: \texttt{black\_hole\_horizon\_radius}  
Code lines: 3
Contained by: module \texttt{black\_hole\_fundamentals}

function: \texttt{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 \texttt{black\_hole\_fundamentals}
Modules used: \texttt{galacticus\_error}

function: \texttt{black\_hole\_horizon\_radius\_spin}
18.1. Program units

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
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:

$$s = -S \frac{P_{\text{rotation}}}{M_\bullet c^2},$$

(18.1)

where $P_{\text{rotation}}$ is the power of rotational energy extraction and

$$S = [(1 + \sqrt{1 - j^2})^2 + j^2] \frac{\sqrt{1 - j^2}}{j},$$

(18.2)

for black hole spin $j$.

**Code lines:** 30

**Contained by:** module `black_hole_fundamentals`

**interface:** `black_hole_static_radius`
18. Source Code Documentation

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.

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.

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].
18.1. Program units

**Description:** Implements calculations of chemical reaction rates for hydrogen using the fits from Abel et al. [1997] and Tegmark et al. [1997].

**Code lines:** 1483

**Contained by:** file `chemical.reaction_rates.hydrogen.F90`

**Modules used:** `chemical_abundances_structure`

**Used by:** subroutine `chemical_reaction_rate` subroutine `chemical_reaction_rates_-initialize`

**subroutine:** `chemical_hydrogen_rate_h2-electron_to_2h-electron`

**Description:** Computes the rate (in units of $c^{-3} s^{-1}$) for the reaction $H_2 + e^- \rightarrow 2H + e^-$.  

**Code lines:** 44

**Contained by:** module `chemical_hydrogen_rates`

**Modules used:** `physical numerical_constants_units` `radiation_structure`

**subroutine:** `chemical_hydrogen_rate_h2-gamma_to_2h`

**Description:** Computes the rate (in units of $cm^{-3} s^{-1}$) for the reaction $H_2^+ + \gamma \rightarrow 2H^+ + e^-$.  

**Code lines:** 49

**Contained by:** module `chemical_hydrogen_rates`

**Modules used:** `physical numerical_constants_units` `radiation_structure`

**subroutine:** `chemical_hydrogen_rate_h2-gamma_to_h2plus_electron`

**Description:** Computes the rate (in units of $cm^{-3} s^{-1}$) for the reaction $H_2 + \gamma \rightarrow H_2^+ + e^-$.  

**Code lines:** 48

**Contained by:** module `chemical_hydrogen_rates`

**Modules used:** `physical numerical_constants_units` `radiation_structure`

**subroutine:** `chemical_hydrogen_rate_h2-gamma_to_h2star_to_2h`

**Description:** Computes the rate (in units of $cm^{-3} s^{-1}$) for the reaction $H_2 + \gamma \rightarrow H_2^* \rightarrow 2H$.  

**Code lines:** 46

**Contained by:** module `chemical_hydrogen_rates`

**Modules used:** `physical numerical_constants_units` `radiation_structure`

**subroutine:** `chemical_hydrogen_rate_h2-h_to_3h`

**Description:** Computes the rate (in units of $c^{-3} s^{-1}$) for the reaction $H_2 + H \rightarrow 3H$.  

**Code lines:** 54

**Contained by:** module `chemical_hydrogen_rates`

**Modules used:** `physical numerical_constants_units` `radiation_structure`

**subroutine:** `chemical_hydrogen_rate_h2-hplus_to_h2plus_h`

**Description:** Computes the rate (in units of $c^{-3} s^{-1}$) for the reaction $H_2 + H^+ \rightarrow H_2^+ + H$.  

**Code lines:** 53

**Contained by:** module `chemical_hydrogen_rates`

**Modules used:** `physical numerical_constants_units` `radiation_structure`
subroutine: chemical_hydrogen_rate_h2plus_electron_to_2h
Description: Computes the rate (in units of c\(^{-3}\) s\(^{-1}\)) for the reaction H\(_2^+\) + e\(^-\) → 2H.
Code lines: 47
Contained by: module chemical_hydrogen_rates
Modules used: radiation_structure

subroutine: chemical_hydrogen_rate_h2plus_gamma_to_2hplus_electron
Description: Computes the rate (in units of cm\(^{-3}\) s\(^{-1}\)) for the reaction H\(_2^+\) + γ → 2H\(^+\) + e\(^-\).
Code lines: 50
Contained by: module chemical_hydrogen_rates
Modules used: numerical_constants_physical numerical_constants_units radiation_structure

subroutine: chemical_hydrogen_rate_h2plus_gamma_to_h_hplus
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_collisional 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
18.1. Program units

**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 cm$^{-3}$ 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`
subroutine: chemical_hydrogen_rate_hplus_electron_to_h_photon
Description: Computes the rate (in units of c^{-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
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 + γ → 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_h2_gamma_to_h2plus_electron
Description: Compute the cross-section (in units of cm^2) for the reaction H_2 + γ → H_2^+ + e^- as given by Abel et al. [1997].
\(^a\)Abel et al. [1997] cite “O’Neil & Reinhardt (1978)” as the source for this fit, but it is not listed in their bibliography, and I have not been able to locate by any other means.
Code lines: 25
Contained by: module chemical_hydrogen_rates
Modules used: numerical_constants_physical numerical_constants_units

function: cross_section_h2plus_gamma_to_2hplus_electron
Description: Compute the cross-section (in units of cm^2) for the reaction H_2^+ + γ → 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_h2plus_gamma_to_h_hplus
Description: Compute the cross-section (in units of cm^2) for the reaction H_2^+ + γ → H + H^+ as given by Shapiro and Kang [1987].
Code lines: 21
Contained by: module chemical_hydrogen_rates
18.1. Program units

**Functions**

### 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:** numerical_constants_physical, numerical_constants_units

### cross_section_hminus_gamma_to_h_electron
- **Description:** Compute the cross-section (in units of cm\(^2\)) for the reaction \(H^- + \gamma \rightarrow H + e^-\) 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
- **Modules used:** numerical_constants_physical, numerical_constants_units

### h_electron_to_hminus_photon_rate_coefficient
- **Description:** Computes the rate coefficient (in units of cm\(^3\) s\(^{-1}\)) for the reaction \(H + e^- \rightarrow H^- + \gamma\).
- **Code lines:** 29
- **Contained by:** module chemical_hydrogen_rates

### h_hminus_to_h2_electron_rate_coefficient
- **Description:** Computes the rate coefficient (in units of cm\(^3\) s\(^{-1}\)) for the reaction \(H + H^- \rightarrow H_2 + e^-\).
- **Code lines:** 21
- **Contained by:** module chemical_hydrogen_rates
- **Modules used:** numerical_constants_physical, numerical_constants_units

### hminus_electron_to_h_2electron_rate_coefficient
- **Description:** Compute the rate coefficient (in units of cm\(^3\) s\(^{-1}\)) for the reaction \(H^- + H \rightarrow H_2 + H^+\).
- **Code lines:** 15
- **Contained by:** module chemical_hydrogen_rates
- **Modules used:** numerical_constants_physical, numerical_constants_units

### hminus_hplus_to_2h_rate_coefficient
- **Description:** Compute the rate coefficient (in units of cm\(^3\) s\(^{-1}\)) for the reaction \(H_2^+ + H \rightarrow H_2 + H^+\).
- **Code lines:** 7
- **Contained by:** module chemical_hydrogen_rates

### 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
18. Source Code Documentation

**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:** Compute 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:** subroutine get_chemical_masses function cooling_radius_isothermal subroutine cooling_radius_solver_null_initialize 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 \(\text{cm}^{-3}\) 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:** 680

**module:** chemical_states_cie_file
**Description:** Reads and interpolates a collisional ionization equilibrium chemical state from a file.
**Code lines:** 660
**Contained by:** file chemical.state.CIE_file.F90
**Modules used:** fgsl iso_varying_string
18.1. Program units

Used by: subroutine chemical_state_initialize subroutine chemical_densities_atomic_cie_cloudy
subroutine chemical_state_atomic_cie_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: 87
Contained by: module chemical_states_cie_file
Modules used: abundances_structure chemical_abundances_structure io_xml 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

641
subroutine: chemical_state_cie_file_read_initialize
Description: Ensure that the cooling data file has been read in.
Code lines: 20
Contained by: module chemicalStates_cie_file

function: do_interpolation
Description: Perform the interpolation.
Code lines: 14
Contained by: module chemicalStates_cie_file

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 chemicalStates_cie_file
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: 76
Contained by: module chemicalStates_cie_file
Modules used: abundances_structure io_xml 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: 79
Contained by: module chemicalStates_cie_file
Modules used: abundances_structure io_xml 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 chemicalStates_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 chemicalStates_cie_file
Modules used: abundances_structure radiation_structure

subroutine: get_interpolation
Description: Determine the interpolating parameters.
Code lines: 29
18.1. Program units

**Contained by:** module chemical_states_cie_file

**Modules used:** 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
dis�_varying_string radiation_structure

**Used by:** subroutine get_chemical_masses subroutine cooling_function cmb_compton

**description:** subroutine cooling_function_density_slope cmb_compton

**function:** cooling_time_simple 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$^{-3}$. 
18.1. Program units

**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  
**Contained by:** file chemical.state.atomic_CIE_cloudy.F90

**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 $\text{cm}^{-3}$.

**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
18. Source Code Documentation

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
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.cooling_function.CIE_file.F90
Description: Contains a module which reads and interpolates a collisional ionization equilibrium cooling function from a file.
Code lines: 571
module: cooling_functions_cie_file
  Description: Reads and interpolates a collisional ionization equilibrium cooling function from a file.
  Code lines: 551
  Contained by: file cooling.cooling_function.CIE_file.F90
  Modules used: fgsl
  Used by: function cooling_function
           subroutine cooling_function_initialize
           subroutine cooling_function_atomic_cie_cloudy
           subroutine cooling_function_temperature_slope_atomic_cie_cloudy
  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
    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: 75
    Contained by: module cooling_functions_cie_file
    Modules used: abundances_structure io_xml
                  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: 79
    Contained by: module cooling_functions_cie_file
    Modules used: abundances_structure io_xml
                  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:**  
- `foxdom`  
- `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:** 13  
**Contained by:** module `cooling_functions_cie_file`

**subroutine: get_interpolation**  
**Description:** Determine the interpolating parameters.  
**Code lines:** 29  
**Contained by:** module `cooling_functions_cie_file`  
**Modules used:** `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`
18.1. Program units

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_states
numerical_constants_units

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_states

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_states

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
iso_varying_string

Used by: function cooling_time_density_log_slope_simple
function cooling_time_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
18.1. Program units

**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 \(\frac{d \ln \Lambda}{d \ln \rho}\) 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`
- `cooling_functions_cmb_compton`
- `cooling_functions_molecular_hydrogen_galli_palla`
- `galacticus_error`
- `input_parameters`
- `memory_management`

**subroutine:** `cooling_function_not_matched`

**Code lines:** 44

**Contained by:** module *cooling_functions*

**Modules used:**
- `galacticus_display`
- `iso_varying_string`
- `string_handling`

**function:** `cooling_function_temperature_log_slope`

**Description:** Return \(\frac{d \ln \Lambda}{d \ln T}\) 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`
- `cooling_functions_cie_file`
- `cooling_functions_cmb_compton`
- `cooling_functions_molecular_hydrogen_galli_palla`

**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`
18. Source Code Documentation

**Used by:**
- function `cooling_function`
- subroutine `cooling_function_initialize`
- function `cooling_function_density_log_slope`
- function `cooling_function_temperature_log_slope`

**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`
- `chemical_abundances_structure`
- `cooling_functions_cie_file`
- `radiation_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`
- `cooling_functions_cie_file`
- `galacticus_input_paths`
- `string_handling`
- `system_command`

**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`
- `cooling_functions_cie_file`
- `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`
18.1. Program units

**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_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  
**Contained by:** module cooling_functions_molecular_hydrogen_galli_palla  
**Modules used:** chemical_abundances_structure

**function:** cooling_function_gp_h_h2  
**Description:** Compute the cooling function due to H–H$_2$ interactions.  
**Code lines:** 23  
**Contained by:** module cooling_functions_molecular_hydrogen_galli_palla  
**Modules used:** chemical_abundances_structure

**function:** cooling_function_gp_h_h2plus  
**Description:** Compute the cooling function due to H–H$_2^+$ interactions.  
**Code lines:** 28  
**Contained by:** module cooling_functions_molecular_hydrogen_galli_palla  
**Modules used:** chemical_abundances_structure

**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$_2^+$ 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  
**Contained by:** module cooling_functions_molecular_hydrogen_galli_palla  
**Modules used:** abundances_structure  
chemical_abundances_structure  
radiation_structure

**subroutine:** cooling_function_molecular_hydrogen_gp_initialize  
**Description:** Initializes the “molecular hydrogen Galli & Palla” cooling function module.  
**Code lines:** 21  
**Contained by:** module cooling_functions_molecular_hydrogen_galli_palla  
**Modules used:** chemical_abundances_structure
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
Contained by: module cooling_functions_molecular_hydrogen_galli_palla
Modules used: abundances_structure chemical_abundances_structure numerical_constants_prefixes radiation_structure

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
Contained by: module cooling_functions_molecular_hydrogen_galli_palla
Modules used: numerical_constants_prefixes

file: cooling.cooling_radius.F90
Description: Contains a module that implements calculations of the cooling radius.
Code lines: 242

module: cooling_radii
Description: Implements calculations of the cooling radius.
Code lines: 222
Contained by: file cooling.cooling_radius.F90
Modules used: cooling_radii_isothermal galacticus_nodes iso_varying_string
Used by: function infall_radius_cooling_freefall function infall_radius_growth_rate_cooling_freefall
function infall_radius_cooling_radius function infall_radius_growth_rate_cooling_radius
subroutine count_properties subroutine establish_property_names
subroutine galacticus_merger_tree_output subroutine node_component_black_hole_simple_rate_compute
function hot_mode_fraction

function: cooling_radius
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
18.1. Program units

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
Modules used: galacticus_error input_parameters

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: 203

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: 181
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: 40
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*: 65  
*Contained by*: module cooling_radii_isothermal  
*Modules used*: chemical_reaction_rates_utilities cooling_times  
cooling_times_available dark_matter_halo_scales  
hot_halo_density_profile 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*: 283

**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*: 262  
*Contained by*: file cooling.cooling_radius.simple.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_simple  
*Description*: Return the growth rate of the cooling radius in the “simple” model in Mpc/Gyr.  
*Code lines*: 65  
*Contained by*: module cooling_radii_simple  
*Modules used*: cooling_times cooling_times_available  
hot_halo_density_profile 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*: 17  
*Contained by*: module cooling_radii_simple  
*Modules used*: cooling_times hot_halo_density_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: 19
Contained by: module cooling_radii_simple
Modules used: galacticus_error iso_varying_string

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: 30
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: 86

module: cooling_rates_cole2000
Description: Implements a Cole et al. [2000] cooling rate calculation.
Code lines: 66
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: 41
Contained by: module cooling_rates_cole2000
Modules used: cooling_infall_radii galacticus_nodes hot_halo_density_profile 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
18. Source Code Documentation

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 galacticus_merger_tree_output
- subroutine node_component_hot_halo_standard_cooling_rate
- subroutine node_component_hot_halo_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 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: 108

module: cooling_rates_white_frenk
  Description: Implements a White and Frenk [1991] cooling rate calculation.
  Code lines: 88
  Contained by: file cooling.cooling_rate.White-Frenk.F90
  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: 39
  Contained by: module cooling_rates_white_frenk
  Modules used: cooling_infall_radii dark_matter_halo_scales
                hot_halo_density_profile 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: 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: 133

module: cooling_rates_modifier_cut_off
  Description: Implements a cut off in the cooling rate at given redshift and virial velocity.
  Code lines: 113
  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: 90
  Contained by: module cooling_rates_modifier_cut_off
18. Source Code Documentation

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: 79

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: 58  
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: 29  
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: 110

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: 89  
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: 16  
Contained by: module cooling_rates_simple_scaling  
Modules used: cosmology_functions
18.1. Program units

**subroutine:** cooling_rate_simple_scaling_initialize  
*Description:* Initializes the “simple scaling” cooling rate module.  
*Code lines:* 53  
*Contained by:* module cooling_rates_simple_scaling  
*Modules used:* 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_-

**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
18. Source Code Documentation

**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`

**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 \( \frac{d \ln t_{\text{cool}}}{d \ln \rho} \) 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 chemical_states cooling_functions numerical_constants_astronomical numerical_constants_physical 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 \( \frac{d \ln t_{\text{cool}}}{d \ln T} \) for gas at the given temperature (in Kelvin), density (in \( M_\odot \) \( \text{Mpc}^{-3} \)), 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`, `iso_varying_string`

*Used by:* function `infall_radius_cooling_freefall`, function `infall_radius_growth_rate_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:* 85
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: 64
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: 15
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
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: 18
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_formation galacticus_nodes

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
18.1. Program units

Function: `cooling_freefall_time_available_get_template`
- Code lines: 5
- Contained by: module `cooling_freefall_times_available`

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
- Contained by: module `cooling_freefall_times_available`

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_error` `iso_varying_string`
- Used by: module `cooling_freefall_times_available`

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_radius
infall_radii_cooling_freefall
iso_varying_string
Used by: function cooling_rate_cole2000
function cooling_rate_white_frenk
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: 107

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: 86
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: 28
Contained by: module infall_radii_cooling_freefall
Modules used: cooling_radii
dark_matter_halo_scales
freefall_radii
galacticus_nodes
18.1. Program units

**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:* 28

*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_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

**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_specificAngular_momentum.F90

*Description:* Contains a module that implements calculations of the specific angular momentum of cooling gas.

*Code lines:* 110

**module:** cooling_specificAngular_momenta
18. Source Code Documentation

**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_constant_rotation`
- `cooling_specific-angular_momenta_mean`
- `galacticus_nodes`
- `iso_varying_string`

**Used by:**
- subroutine `node_component_hot_halo_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

**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:** 192

**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:** 171

**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:** 62
18.1. Program units

**Contained by:** module `cooling_specific_angular_momenta_constant_rotation`

**Modules used:**
- `dark_matter_profiles`
- `galacticus_nodes`
- `hot_halo_density_profile`
- `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:** 64

**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:** 43

- **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:** 15

- **Contained by:** module `cooling_specific-angular-momenta-mean`

**function:** `cooling_specific-angular-momentum-mean`

- **Description:** Return the specific angular momentum of cooling gas in the mean model.
- **Code lines:** 12

- **Contained by:** module `cooling_specific-angular-momenta-mean`

**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`
  - function `cooling_radius_isothermal`
  - function `cooling_radius_growth-rate-simple`
  - function `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: 102

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: 81
Contained by: file cooling.time_available.White-Frenk.F90
Used by: module cooling_times_available

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: 23
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.
18.1. Program units

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
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 implements useful cosmological functions.
Code lines: 468

module: cosmology_functions
Description: Implements useful cosmological functions.
Code lines: 448
Contained by: file cosmology.functions.F90
Modules used: iso_c_binding iso_varying_string
Used by: program tests_excursion_sets program optimal_sampling_smf subroutine accretion_halos_simple_-initialize subroutine cooling_rate_modifier_cut_-off
18.1. Program units

subroutine excursion_sets_first_crossing_rate_tabulate_farahi
function linear_growth_factor
module linear_growth_simple
function power_spectrum_nonlinear_peacockdodds1996
subroutine perturbation_dynamics_solver
function halo_virial_density_contrast_rate_of_change
subroutine virial_density_kitayama_suto1996
program test_nfw96_concentration_dark_energy
program test_zhao2009_flat
program test_zhao2009_open
program tests_comoving_distance_dark_energy
program tests_comoving_distance_open
program tests_cosmic_age_eds
program tests_cosmic_age_dark_energy_closed
program tests_cosmic_age_dark_energy_omega_minus_one_third
program tests_halo_mass_function_tinker
program tests_linear_growth_cosmological_constant
program tests_linear_growth_energy
program tests_linear_growth_open
program tests_sphericalCollapse_dark_energy_omega_zero_point_six
program tests_sphericalCollapse_dark_energy_omega_zero_point_eight
program tests_sphericalCollapse_dark_energy_omega_half
program tests_sphericalCollapse_dark_energy_lambda
program tests_sphericalCollapse_dark_energy_open
program tests_sphericalCollapse_flat

function: cmb_temperature
Description: Return the temperature of the cosmic microwave background at aExpansion.
Code lines: 13
Contained by: module cosmology_functions

function: comoving_distance
Description: Return the comoving distance to the given cosmic time.
Code lines: 12
18. Source Code Documentation

**Contained by:** module `cosmology_functions`

**function:** comoving_distance_conversion
**Description:** Convert between different measures of comoving distance.
**Code lines:** 13
**Contained by:** module `cosmology_functions`

**function:** comoving_volume_element_redshift
**Description:** 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.
**Code lines:** 10
**Contained by:** module `cosmology_functions`
**Modules used:** numerical_constants_physical

**function:** comoving_volume_element_time
**Description:** 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.
**Code lines:** 11
**Contained by:** module `cosmology_functions`
**Modules used:** numerical_constants_astronomical numerical_constants_physical

**function:** cosmic_time_is_valid
**Description:** Returns true if the given cosmic time is valid one for this cosmology.
**Code lines:** 12
**Contained by:** module `cosmology_functions`

**function:** cosmology_age
**Description:** Return the age of the universe (in Gyr) at given expansion factor.
**Code lines:** 13
**Contained by:** module `cosmology_functions`

**function:** cosmology_dark_energy_equation_of_state
**Description:** Return the dark energy equation of state.
**Code lines:** 10
**Contained by:** module `cosmology_functions`

**function:** cosmology_dark_energy_exponent
**Description:** Return the dark energy equation of state.
**Code lines:** 10
**Contained by:** module `cosmology_functions`

**subroutine:** cosmology_functions_initialize
**Description:** Initialize the cosmology functions module.
**Code lines:** 54
**Contained by:** module `cosmology_functions`
**Modules used:** cosmology_functions_matter_dark_energy cosmology_functions_matter_lambda

674
18.1. Program units

```
subroutine: early_time_density_scaling
  Description: Compute the scaling of density with expansion factor at early times in the universe.
  Code lines: 14
  Contained by: module cosmology_functions

function: epoch_of_matter_curvature_equality
  Description: Return the epoch of matter-curvature magnitude equality (either expansion factor or cosmic time).
  Code lines: 12
  Contained by: module cosmology_functions

function: epoch_of_matter_dark_energy_equality
  Description: Return the epoch of matter-dark energy magnitude equality (either expansion factor or cosmic time).
  Code lines: 12
  Contained by: module cosmology_functions

function: epoch_of_matter_domination
  Description: Compute the epoch at which matter dominates over other forms of energy by a given factor.
  Code lines: 12
  Contained by: module cosmology_functions

function: expansion_factor
  Description: Returns the expansion factor at cosmological time $t_{\text{Cosmological}}$.
  Code lines: 12
  Contained by: module cosmology_functions

function: expansion_factor_c
  Description: A C-bound wrapper function for Expansion_Factor().
  Code lines: 8
  Contained by: module cosmology_functions

function: expansion_factor_from_redshift
  Description: Returns expansion factor given a redshift.
  Code lines: 7
  Contained by: module cosmology_functions

function: expansion_factor_is_valid
  Description: Returns true if the given expansion factor is valid one for this cosmology.
  Code lines: 12
  Contained by: module cosmology_functions

function: expansion_rate
  Description: Returns the cosmological expansion rate, $\dot{a}/a$ at expansion factor $a_{\text{Expansion}}$.
  Code lines: 12
  Contained by: module cosmology_functions
```
function: hubble_parameter
Description: Returns the Hubble parameter at the request cosmological time, tCosmological, or expansion factor, aExpansion.
Code lines: 13
Contained by: module cosmology_functions

function: omega_dark_energy
Description: Return the dark energy density parameter at expansion factor aExpansion.
Code lines: 13
Contained by: module cosmology_functions

function: omega_matter_total
Description: Return the matter density parameter at expansion factor aExpansion.
Code lines: 13
Contained by: module cosmology_functions

function: redshift_from_expansion_factor
Description: Returns redshift for a given expansion factor.
Code lines: 7
Contained by: module cosmology_functions

function: time_from_comoving_distance
Description: Return the cosmic time corresponding to the given comovingDistance.
Code lines: 12
Contained by: module cosmology_functions

file: cosmology.functions.matter_dark_energy.F90
Description: Contains a module which implements useful cosmological functions. This implementation assumes a Universe filled with 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: 808

module: cosmology_functions_matter_dark_energy
Description: Implements useful cosmological functions. This implementation assumes a Universe filled with 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: 786
Contained by: file cosmology.functions.matter_dark_energy.F90
Modules used: cosmological_parameters, fgsl, iso_c_binding
Used by: subroutine cosmology_functions_, subroutine galacticus_state_retrieve
initialize
subroutine galacticus_state_store

function: agetableodesdarkenergy
Description: System of differential equations to solve for expansion factor vs. age.
Code lines: 10
Contained by: module cosmology_functions_matter_dark_energy
18.1. Program units

function: cmb_temperature_matter_dark_energy
Description: Return the temperature of the CMB at expansion factor \( a_{\text{Expansion}} \).
Code lines: 24
Contained by: module cosmology_functions_matter_dark_energy
Modules used: galacticus_error

function: comoving_distance_conversion_matter_dark_energy
Description: Convert between different measures of distance.
Code lines: 9
Contained by: module cosmology_functions_matter_dark_energy
Modules used: galacticus_error

function: comoving_distance_matter_dark_energy
Description: Returns the comoving distance to cosmological time \( t_{\text{Cosmological}} \).
Code lines: 8
Contained by: module cosmology_functions_matter_dark_energy
Modules used: galacticus_error

function: cosmic_time_is_valid_matter_dark_energy
Description: Checks that the time falls within allowed ranges.
Code lines: 7
Contained by: module cosmology_functions_matter_dark_energy

function: cosmology_age_matter_dark_energy
Code lines: 41
Contained by: module cosmology_functions_matter_dark_energy
Modules used: galacticus_error numerical_interpolation

function: cosmology_dark_energy_equation_of_state_dark_energy
Description: Return the dark energy equation of state.
Code lines: 17
Contained by: module cosmology_functions_matter_dark_energy
Modules used: galacticus_error

function: cosmology_dark_energy_exponent_dark_energy
Description: Return the dark energy equation of state.
Code lines: 21
Contained by: module cosmology_functions_matter_dark_energy
Modules used: galacticus_error

subroutine: cosmology_functions_matter_dark_energy_initialize
Description: Initialize the module.
Code lines: 73
Contained by: module cosmology_functions_matter_dark_energy
Modules used: input_parameters iso_varying_string

subroutine: cosmology_matter_dark_energy_state_retrieve
18. Source Code Documentation

**Description:** Retrieve the tabulation state from the file.

**Code lines:** 19

**Contained by:** module `cosmology_functions_matter_dark_energy`

**Modules used:** `memory_management`

**subroutine:** `cosmology_matter_dark_energy_state_store`

**Description:** Write the tabulation state to file.

**Code lines:** 11

**Contained by:** module `cosmology_functions_matter_dark_energy`

**subroutine:** `early_time_density_scaling_matter_dark_energy`

**Code lines:** 15

**Contained by:** module `cosmology_functions_matter_dark_energy`

**function:** `epoch_of_matter_curvature_equality_matter_dark_energy`

**Description:** Return the epoch of matter-curvature magnitude equality (either expansion factor or cosmic time).

**Code lines:** 16

**Contained by:** module `cosmology_functions_matter_dark_energy`

**Modules used:** `cosmology_functions_parameters`  

**function:** `epoch_of_matter_dark_energy_equality_matter_dark_energy`

**Description:** Return the epoch of matter-dark energy magnitude equality (either expansion factor or cosmic time).

**Code lines:** 51

**Contained by:** module `cosmology_functions_matter_dark_energy`

**Modules used:** `cosmology_functions_parameters`  

**function:** `epoch_of_matter_domination_matter_dark_energy`

**Code lines:** 62

**Contained by:** module `cosmology_functions_matter_dark_energy`

**Modules used:** `cosmology_functions_parameters`  

**root_finder**

**function:** `expansion_factor_change`

**Description:** Compute the expansion factor at time `timeEnd` given an initial value `expansionFactorStart` at time `timeStart`.

**Code lines:** 16

**Contained by:** module `cosmology_functions_matter_dark_energy`

**Modules used:** `ode_solver`

**function:** `expansion_factor_is_valid_matter_dark_energy`

**Description:** Checks that the expansion factor falls within allowed ranges.

**Code lines:** 7

**Contained by:** module `cosmology_functions_matter_dark_energy`

**function:** `expansion_factor_matter_dark_energy`

**Description:** Returns the expansion factor at cosmological time `tCosmological`.

**Code lines:** 46

**Contained by:** module `cosmology_functions_matter_dark_energy`
18.1. Program units

Modules used: galacticus_error   numerical_interpolation

function: expansion_rate_matter_dark_energy
Description: Returns the cosmological expansion rate, \( \dot{a}/a \) at expansion factor \( a_{\text{Expansion}} \).
Code lines: 8
Contained by: module cosmology_functions_matter_dark_energy

function: hubble_parameter_matter_dark_energy
Description: Returns the Hubble parameter at the request cosmological time, \( t_{\text{Cosmological}} \), or expansion factor, \( a_{\text{Expansion}} \).
Code lines: 36
Contained by: module cosmology_functions_matter_dark_energy

subroutine: make_expansion_factor_table
Description: Builds a table of expansion factor vs. time.
Code lines: 123
Contained by: module cosmology_functions_matter_dark_energy
Modules used: memory_management   numerical_interpolation   numerical_ranges

function: matter_dark_energy_domination
Code lines: 6
Contained by: module cosmology_functions_matter_dark_energy

function: omega_dark_energy_matter_dark_energy
Description: Return the dark energy density parameter at expansion factor \( a_{\text{Expansion}} \).
Code lines: 24
Contained by: module cosmology_functions_matter_dark_energy
Modules used: galacticus_error

function: omega_matter_total_matter_dark_energy
Description: Return the matter density parameter at expansion factor \( a_{\text{Expansion}} \).
Code lines: 24
Contained by: module cosmology_functions_matter_dark_energy
Modules used: galacticus_error

function: time_from_comoving_distance_matter_dark_energy
Description: Returns the cosmological time corresponding to given \( \text{comovingDistance} \).
Code lines: 8
Contained by: module cosmology_functions_matter_dark_energy
Modules used: galacticus_error

file: cosmology.functions.matter_lambda.F90
Description: Contains a module which implements useful cosmological functions. This implementation assumes a Universe filled with collisionless matter and a cosmological constant.
Code lines: 891

module: cosmology_functions_matter_lambda
18. Source Code Documentation

**Description:** Implements useful cosmological functions. This implementation assumes a Universe filled with collisionless matter and a cosmological constant.

**Code lines:** 870

**Contained by:** file `cosmology.functions.matter_lambda.F90`

**Modules used:**
- `cosmological_parameters`
- `fgsl`
- `iso_c_binding`

**Used by:**
- subroutine `cosmology_functions_initialize`
- subroutine `galacticus_state_retrieve initial`
- subroutine `galacticus_state_store`

---

**function:** `agetableodes`

**Description:** System of differential equations to solve for expansion factor vs. age.

**Code lines:** 10

**Contained by:** module `cosmology_functions_matter_lambda`

---

**function:** `cmb_temperature_matter_lambda`

**Description:** Return the temperature of the CMB at expansion factor $a_{\text{Expansion}}$.

**Code lines:** 24

**Contained by:** module `cosmology_functions_matter_lambda`

**Modules used:** `galacticus_error`

---

**function:** `collapseodes`

**Description:** System of differential equations to solve for age vs. expansion factor.

**Code lines:** 10

**Contained by:** module `cosmology_functions_matter_lambda`

---

**function:** `comoving_distance_conversion_matter_lambda`

**Description:** Convert between different measures of distance.

**Code lines:** 43

**Contained by:** module `cosmology_functions_matter_lambda`

**Modules used:**
- `cosmology_functions_options`
- `galacticus_error`
- `numerical_interpolation`

---

**function:** `comoving_distance_integrand`

**Description:** Integrand function used in computing the comoving distance.

**Code lines:** 11

**Contained by:** module `cosmology_functions_matter_lambda`

**Modules used:**
- `numerical_constants_astronomical`
- `numerical_constants_physical`

---

**function:** `comoving_distance_matter_lambda`

**Description:** Returns the comoving distance to cosmological time $t_{\text{Cosmological}}$.

**Code lines:** 28

**Contained by:** module `cosmology_functions_matter_lambda`

**Modules used:**
- `galacticus_error`
- `numerical_interpolation`

---

**function:** `cosmic_time_is_valid_matter_lambda`

**Description:** Checks that the time falls within allowed ranges.

**Code lines:** 7

**Contained by:** module `cosmology_functions_matter_lambda`
function: cosmology_age_matter_lambda
   Code lines:    46
   Contained by: module cosmology_functions_matter_lambda
   Modules used: galacticus_error             numerical_interpolation

function: cosmology_dark_energy_equation_of_state_matter_lambda
   Description: Return the dark energy equation of state.
   Code lines:    7
   Contained by: module cosmology_functions_matter_lambda

function: cosmology_dark_energy_exponent_matter_lambda
   Description: Return the dark energy equation of state.
   Code lines:    7
   Contained by: module cosmology_functions_matter_lambda

subroutine: cosmology_functions_matter_lambda_initialize
   Description: Initialize the module.
   Code lines:   112
   Contained by: module cosmology_functions_matter_lambda
   Modules used: iso_varying_string             numerical_comparison
                        ode_solver

subroutine: cosmology_matter_lambda_state_retrieve
   Description: Retrieve the tabulation state from the file.
   Code lines:    30
   Contained by: module cosmology_functions_matter_lambda
   Modules used: memory_management

subroutine: cosmology_matter_lambda_state_store
   Description: Write the tablulation state to file.
   Code lines:    13
   Contained by: module cosmology_functions_matter_lambda

subroutine: early_time_density_scaling_matter_lambda
   Code lines:    15
   Contained by: module cosmology_functions_matter_lambda

function: epoch_of_matter_curvature_equality_matter_lambda
   Description: Return the epoch of matter-curvature magnitude equality (either expansion factor or cosmic time).
   Code lines:    16
   Contained by: module cosmology_functions_matter_lambda
   Modules used: cosmology_functions_parameters

function: epoch_of_matter_dark_energy_equality_matter_lambda
   Description: Return the epoch of matter-dark energy magnitude equality (either expansion factor or cosmic time).
   Code lines:    16
   Contained by: module cosmology_functions_matter_lambda
18. Source Code Documentation

Modules used: cosmology_functions_parameters

function: epoch_of_matter_domination_matter_lambda
  Code lines: 33
  Contained by: module cosmology_functions_matter_lambda
  Modules used: cosmology_functions_parameters

function: expansion_factor_is_valid_matter_lambda
  Description: Checks that the expansion factor falls within allowed ranges.
  Code lines: 7
  Contained by: module cosmology_functions_matter_lambda

function: expansion_factor_matter_lambda
  Description: Returns the expansion factor at cosmological time tCosmological.
  Code lines: 45
  Contained by: module cosmology_functions_matter_lambda
  Modules used: galacticus_error numerical_interpolation

function: expansion_rate_matter_lambda
  Description: Returns the cosmological expansion rate, \( \dot{a}/a \) at expansion factor aExpansion.
  Code lines: 8
  Contained by: module cosmology_functions_matter_lambda

function: hubble_parameter_matter_lambda
  Description: Returns the Hubble parameter at the request cosmological time, tCosmological, or expansion factor, aExpansion.
  Code lines: 36
  Contained by: module cosmology_functions_matter_lambda
  Modules used: galacticus_error

subroutine: make_distance_table
  Description: Builds a table of distance vs. time.
  Code lines: 56
  Contained by: module cosmology_functions_matter_lambda
  Modules used: memory_management numerical_integration
  numerical_interpolation numerical_ranges

subroutine: make_expansion_factor_table
  Description: Builds a table of expansion factor vs. time.
  Code lines: 98
  Contained by: module cosmology_functions_matter_lambda
  Modules used: memory_management numerical_integration
  numerical_ranges ode_solver

function: omega_dark_energy_matter_lambda
  Description: Return the dark energy density parameter at expansion factor aExpansion.
  Code lines: 24
  Contained by: module cosmology_functions_matter_lambda
  Modules used: galacticus_error
function: omega_matter_total_matter_lambda
Description: Return the matter density parameter at expansion factor aExpansion.
Code lines: 24
Contained by: module cosmology_functions_matter_lambda
Modules used: galacticus_error

function: time_from_comoving_distance_matter_lambda
Description: Returns the cosmological time corresponding to given comovingDistance.
Code lines: 31
Contained by: module cosmology_functions_matter_lambda
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 comoving_distance_conversion_matter_lambda
program tests_comoving_distance_eds
program tests_comoving_distance_dark_energy

file: cosmology.functions.parameters.F90
Description: Contains a module which option parameters for cosmology functions.
Code lines: 29

module: cosmology_functions_parameters
Description: Defines option parameters for cosmology functions.
Code lines: 9
Contained by: file cosmology.functions.parameters.F90
Used by: function epoch_of_matter_curvature_matter_lambda
function epoch_of_matter_dark_energy_matter_lambda
function epoch_of_matter_domination_matter_lambda

file: cosmology.parameters.F90
Description: Contains a module which handles cosmological parameters.
Code lines: 265

module: cosmological_parameters
Description: Implements cosmological parameters and related derived quantities. Default parameter values are taken from Hinshaw et al. [2012].
Code lines: 245
18. Source Code Documentation

**Contained by:**
file `cosmology.parameters.F90`

**Modules used:**
- program `tests_excursion_sets`
- function `halo_baryonic_accreted_mass_-simple_get`
- function `halo_baryonic_failed_-accreted_mass_simple_get`
- module `cosmology_functions_matter_-dark_energy`
- function `dark_matter_halo_mean_density`
- function `dark_matter_profile_-concentration_prada2011`
- subroutine `galactic_structure_radii_-initial_adiabatic`
- subroutine `galacticus_output_tree_-density_contrast_initialize`
- subroutine `galacticus_merger_tree_-output_filter_lightcone.initialize`
- function `igm_state_electron_-scattering_integrand`
- subroutine `merger_tree_read_initialize`
- subroutine `merger_tree_write`
- subroutine `node_component_hot_halo_-standard_node_merger`
- subroutine `power_spectrum_compute`
- subroutine `critical_overdensity_-kitayama_suto1996_initialize`
- function `halo_mass_fraction_integrated`
- function `halo_mass_function_sheth_-tormen_differential`
- module `linear_growth_simple`

**Used by:**
- subroutine `get_chemical_masses`
- function `halo_baryonic_accretion_rate_-simple_get`
- function `halo_baryonic_failed_-accretion_rate_simple_get`
- module `cosmology_functions_matter_-lambda`
- function `dark_matter_profile_-concentration_munozcuarticolas2011`
- function `dark_matter_profile_density_-task`
- subroutine `galactic_structure_radii_-solve_adiabatic`
- subroutine `galacticus_output_tree_-density_contrast_initialize`
- function `hot_halo_density_cored_-isothermal_core_radius_growing_core`
- subroutine `intergalactic_medium_state_-refast_initialize`
- subroutine `merger_trees_simple_process`
- subroutine `node_component_black_hole_-standard_rate_compute`
- subroutine `node_component_hot_halo_-very_simple_tree_initialize`
- 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`
- function `power_spectrum_nonlinear_-cosmicemu`
- function `power_spectrum_-function_top_hat`
- subroutine `power_spectrum_window_-functions_th_kss_hybrid_initialize`
- subroutine `transfer_function_bbks_make`
- subroutine `transfer_function_eisenstein_hu_make`
- subroutine `transfer_function_file_read`
18.1. Program units

subroutine virial_density_bryan_norman_initialize
program test_nfw96_concentration_dark_energy
program tests_cosmic_age_dark_energy_closed
program tests_power_spectrum

function: critical_density
Description: Returns the critical density in units of \( M_\odot/\text{Mpc}^3 \).
Code lines: 17
Contained by: module cosmological_parameters
Modules used: numerical_constants_math numerical_constants_physical

function: h_0
Description: Returns the value of \( H_0 \), reading it in first if necessary.
Code lines: 30
Contained by: module cosmological_parameters
Modules used: galacticus_display

function: h_0_invgyr
Description: Returns the value of \( H_0 \), in units of inverse Gyr.
Code lines: 16
Contained by: module cosmological_parameters
Modules used: numerical_constants_astronomical

function: little_h_0
Description: Returns \( h_0 = H_0/100 \text{ km/s/Mpc} \).
Code lines: 7
Contained by: module cosmological_parameters

function: omega_b
Description: Returns the value of \( \Omega_b \), reading it in first if necessary.
Code lines: 26
Contained by: module cosmological_parameters

function: omega_de
Description: Returns the value of \( \Omega_b \), reading it in first if necessary.
Code lines: 26
Contained by: module cosmological_parameters

function: omega_k
Description: Returns the value of \( \Omega_K \), computing it first if necessary.
Code lines: 15
Contained by: module cosmological_parameters

function: omega_matter
Description: Returns the value of \( \Omega_b \), reading it in first if necessary.
Code lines: 31

685
function: omega_radiation  
Description: Returns the value of $\Omega_r$, computing it first if necessary.  
Code lines: 17  
Contained by: module cosmological_parameters  
Modules used: galacticus_error

function: t_cmb  
Description: Returns the value of $T_{\text{CMB}}$, reading it in first if necessary.  
Code lines: 26  
Contained by: module cosmological_parameters

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_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

function: dark_matter_halo_mass_accretion_time  
Description: Returns the time for thisNode in thisTree according to the mass accretion history.
**18.1. Program units**

**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, dark_matter_halo_mass_accretion_histories_wechsler2002, dark_matter_halo_mass_accretion_histories_zhao2009, galacticus_error, input_parameters

**file:** dark_matter_halos.mass_accretion_history.Wechsler2002.F90  
**Description:** Contains a module which implements the Wechsler et al. [2002] halo mass accretion algorithm.  
**Code lines:** 136

**module:** dark_matter_halo_mass_accretion_histories_wechsler2002  
**Description:** Implements the Wechsler et al. [2002] halo mass accretion algorithm.  
**Code lines:** 116  
**Contained by:** file dark_matter_halos.mass_accretion_history.Wechsler2002.F90  
**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:** 30  
**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:** 23  
**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:**
- f gsl
- 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

**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`
18.1. Program units

**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.

**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:** 82

**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:** 61

**Contained by:** file `dark_matter_halos.mass_loss_rates.vanDenBosch.F90`

**Used by:** subroutine `dark_matter_halo_mass_loss_rates_vanDenBoschพนักงานrates_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:** 22

**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.
Source Code Documentation

Code lines: 17
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 implements calculations of various scales for dark matter halos.
Code lines: 278

module: dark_matter_halo_scales
Description: Implements calculations of various scales for dark matter halos.
Code lines: 258
Contained by: file dark_matter_halos.scales.F90
Modules used: galacticus_nodes kind_numbers tables
Used by: subroutine get_chemical_masses function halo_baryonic_accreted_mass_simple_get
function halo_baryonic_accretion_rate_simple_get function halo_baryonic_failed_accreted_mass_simple_get
function halo_baryonic_failed_accretion_rate_simple_get function black_hole_binary_initial_radius_volonteri_2003
function black_hole_binary_separation_growth_rate_standard function cooling_radius_growth_rate_isothermal
function cooling_radius_isothermal function cooling_rate_white_frenk
subroutine cooling_rate_modifier_cutoff function infall_radius_white_frenk
function infall_radius_growth_rate_cooling_freefall
function dark_matter_profile_density_einasto function dark_matter_profile_enclosed_mass_einasto
function dark_matter_profile_energy_einasto function dark_matter_profile_energy_growth_rate_einasto
function dark_matter_profile_kspace_einasto function dark_matter_profile_potential_einasto
function dark_matter_profile_rotation_normalization_einasto function dark_matter_profile_density_nfw
function dark_matter_profile_enclosed_mass_nfw function dark_matter_profile_energy_growth_rate_nfw
function dark_matter_profile_energy_nfw function dark_matter_profile_freefall_radius_increase_rate_nfw
function dark_matter_profile_freefall_radius_nfw function dark_matter_profile_kspace_nfw
function dark_matter_profile_potential_nfw function dark_matter_profile_rotation_normalization_nfw
function dark_matter_profile_circular_velocity_isothermal function dark_matter_profile_density_isothermal
function dark_matter_profile_enclosed_mass_isothermal
function dark_matter_profile_energy_growth_rate_isothermal
function dark_matter_profile_energy_isothermal
function dark_matter_profile_freefall_radius_increase_rate_isothermal
function dark_matter_profile_freefall_radius_isothermal
function dark_matter_profile_potential_isothermal
function radius_from_specific-angular-momentum_isothermal
function dark_matter_profile_energy_isothermal
function dark_matter_profile_energy_isothermal
function dark_matter_profile_freefall_radius_increase_rate_isothermal
function dark_matter_profile_freefall_radius_isothermal
function dark_matter_profile_potential_isothermal
function radius_from_specific-angular-momentum_isothermal
function dark_matter_profile_freefall_radius_increase_rate_isothermal
function dark_matter_profile_freefall_radius_isothermal
function dark_matter_profile_potential_isothermal
function radius_from_specific-angular-momentum_isothermal

subroutine solve_for_radius

subroutine solve_for_radius

subroutine galacticus_output_tree_density_contrast
subroutine galacticus_output_tree_velocity_dispersion
subroutine galacticus_state_retrieve
subroutine halo_mass_function_compute

function hot_halo_density_cored_isothermal_core_radius_virial_fraction
function hot_halo_ram_pressure_stripping_virial_radius
subroutine assign_scale_radii
subroutine scan_for_mergers
subroutine merger_trees_render_dump

function hot_halo_density_cored_isothermal_core_radius_virial_fraction
function hot_halo_ram_pressure_stripping_virial_radius
subroutine assign_scale_radii
subroutine scan_for_mergers
subroutine merger_trees_render_dump

subroutine node_component_black_hole_simple_rate_compute
subroutine node_component_black_hole_standard_rate_compute
subroutine node_component_black_hole_standard_rate_compute
subroutine node_component_disk_exponential_post_evolve

subroutine node_component_disk_standard_rate_compute
subroutine node_component_disk_exponential_radius_solver_plausibility
subroutine node_component_disk_standard_rate_compute
subroutine node_component_disk_exponential_radius_solver_plausibility

function node_component_disk_standardOuterRadius

function node_component_hot_halo_standard_outerRadius

function node_component_hot_halo_standard_outerRadius

subroutine node_component_hot_halo_standard_create
subroutine node_component_hot_halo_standard_create
subroutine node_component_hot_halo_standard_node_merger

function node_component_hot_halo_standard_outflow_stripped_fraction

function node_component_hot_halo_standard_outflow_stripped_fraction

subroutine node_component_hot_halo_standard_formation
subroutine node_component_hot_halo_standard_node_merger

function node_component_hot_halo_standard_outflow_stripped_fraction

function node_component_hot_halo_standard_outflow_stripped_fraction

subroutine node_component_hot_halo_standard_formation
subroutine node_component_hot_halo_standard_node_merger
18. Source Code Documentation

subroutine node_component_hot_halo_- standard_post_evolve
subroutine node_component_hot_halo_- standard_push_to_null
subroutine node_component_hot_halo_- standard_satellite_merger
subroutine node_component_hot_halo_- standard_rate_compute
subroutine node_component_hot_halo_- standard_scale_set

subroutine node_component_merging_- statistics_recent_node_merger

function satellite_time_until_merging_- boylankolchin2008
function satellite_time_until_merging_- jiang2008
function satellite_time_until_merging_- lacey_cole
function satellite_time_until_merging_- lacey_cole_tormen
function satellite_time_until_merging_- size_covington2008
function virial_orbital_parameters_- benson2005
function virial_orbital_parameters_- wetzel2010
function satellite_orbit_equivalent_- circular_orbit_radius
function starFormation_feedback_disk_- outflow_rate_halo_scaling
function starFormation_timescale_- disk_halo_scaling

function: dark_matter_halo_dynamical_timescale
Description: Returns the dynamical timescale for thisNode.
Code lines: 18
Contained by: module dark_matter_halo_scales
Modules used: numerical_constants_astronomical

function: dark_matter_halo_mean_density
Description: Returns the mean density for thisNode.
Code lines: 36
Contained by: module dark_matter_halo_scales
Modules used: cosmological_parameters cosmology_functions virial_density_contrast

function: dark_matter_halo_mean_density_growth_rate
Description: Returns the growth rate of the mean density for thisNode.
Code lines: 31
Contained by: module dark_matter_halo_scales
Modules used: cosmology_functions virial_density_contrast

subroutine: dark_matter_halo_scales_reset
Description: Reset the cooling radius calculation.
Code lines: 11
Contained by: module dark_matter_halo_scales

subroutine: dark_matter_halo_scales_state_retrieve
Description: Retrieve the tabulation state from the file.
Code lines: 11
Contained by: module dark_matter_halo_scales
Modules used: fgsl
18.1. Program units

**subroutine:** dark_matter_halo_scales_state_store
*Description:* Write the tablulation state to file.
*Code lines:* 9
*Contained by:* module dark_matter_halo_scales
*Modules used:* fgsl

**function:** dark_matter_halo_virial_radius
*Description:* Returns the virial radius scale for thisNode.
*Code lines:* 23
*Contained by:* module dark_matter_halo_scales
*Modules used:* numerical_constants_math

**function:** dark_matter_halo_virial_radius_growth_rate
*Description:* Returns the growth rate of the virial radius scale for thisNode.
*Code lines:* 10
*Contained by:* module dark_matter_halo_scales

**function:** dark_matter_halo_virial_temperature
*Description:* Returns the virial temperature (in Kelvin) for thisNode.
*Code lines:* 19
*Contained by:* module dark_matter_halo_scales
*Modules used:* numerical_constants_astronomical numerical_constants_physical

**function:** dark_matter_halo_virial_velocity
*Description:* Returns the virial velocity scale for thisNode.
*Code lines:* 23
*Contained by:* module dark_matter_halo_scales
*Modules used:* numerical_constants_physical

**function:** dark_matter_halo_virial_velocity_growth_rate
*Description:* Returns the growth rate of the virial velocity scale for thisNode.
*Code lines:* 10
*Contained by:* module dark_matter_halo_scales

**file:** dark_matter_halos.spins.F90
*Description:* Contains a module which implements calculations of dark matter halo angular momentum.
*Code lines:* 90

**module:** dark_matter_halo_spins
*Description:* Implements calculations of dark matter halo angular momentum.
*Code lines:* 70
*Contained by:* file dark_matter_halos.spins.F90
*Used by:* 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:* 17
18. Source Code Documentation

**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:* 18

**subroutine: dark_matter_halo_spins_initialize**

*Description:* Initialize the halo spins module.

*Code lines:* 18


*Description:* Contains a module which implements the Bett et al. [2007] halo spin distribution.

*Code lines:* 155

**module: halo_spin_distributions_bett2007**

*Description:* Implements the Bett et al. [2007] halo spin distribution.

*Code lines:* 135

**function: halo_spin_distribution_bett2007**

*Description:* Return a halo spin from a lognormal distribution.

*Code lines:* 11

**subroutine: halo_spin_distribution_bett2007_initialize**

*Description:* Initializes the “Bett2007” halo spin distribution module.

*Code lines:* 49

**subroutine: halo_spin_distribution_bett2007_snapshot**

*Description:* Store a snapshot of the random number generator internal state.

*Code lines:* 7

**subroutine: halo_spin_distribution_bett2007_state_retrieve**

*Description:* Write the stored snapshot of the random number state to file.
18.1. Program units

**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*

**file:** *dark_matter_halos.spins.distributions.F90*  
**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*

**Used by:** subroutine *node_component_spin_random_-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_-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
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: Contains a module which implements Einasto halo profiles.
Code lines: 1075

module: dark_matter_profiles_einasto
Description: Implements Einasto halo profiles.
Code lines: 1055
Contained by: file dark_matter_profiles.Einasto.F90
Modules used: f gsl galacticus_nodes
Used by: subroutine dark_matter_profile_-initialize
dark_matter_state_retrieve
dark_matter_state_store

function: dark_matter_profile_circular_velocity_einasto
Description: Returns the circular velocity (in km/s) in the dark matter profile of thisNode at the given radius (given in units of Mpc).
Code lines: 14
Contained by: module dark_matter_profiles_einasto
Modules used: numerical_constants_physical

function: dark_matter_profile_density_einasto
Description: Returns the density (in $M_\odot Mpc^{-3}$) in the dark matter profile of thisNode at the given radius (given in units of Mpc).
Code lines: 21
Contained by: module dark_matter_profiles_einasto
Modules used: dark_matter_halo_scales

subroutine: dark_matter_profile_einasto_freefall_tabulate
Description: Tabulates the freefall time vs. freefall radius for Einasto halos.
Code lines: 81
Contained by: module dark_matter_profiles_einasto
Modules used: galacticus_display memory_management
tabulation numerical_interpolation numerical_ranges

subroutine: dark_matter_profile_einasto_initialize
Description: Initializes the “Einasto” halo profile module.
Code lines: 36
Contained by: module dark_matter_profiles_einasto
Modules used: galacticus_error iso_varying_string

function: dark_matter_profile_enclosed_mass_einasto
Description: Returns the enclosed mass (in $M_\odot$) in the dark matter profile of thisNode at the given radius (given in units of Mpc).
18. Source Code Documentation

*Code lines:* 21  
*Contained by:* module `dark_matter_profiles_einasto`  
*Modules used:* `dark_matter_halo_scales`

**function:** `dark_matter_profile_energy_einasto`  
*Description:* Return the energy of an Einasto halo density profile.  
*Code lines:* 41  
*Contained by:* module `dark_matter_profiles_einasto`  
*Modules used:* `dark_matter_halo_scales`  
**function:** `dark_matter_profile_energy_growth_rate_einasto`  
*Description:* Return the energy of an Einasto halo density profile.  
*Code lines:* 44  
*Contained by:* module `dark_matter_profiles_einasto`  
*Modules used:* `dark_matter_halo_scales`  
**function:** `dark_matter_profile_freefall_radius_einasto`  
*Description:* Returns the freefall radius in the Einasto density profile at the specified `time` (given in Gyr).  
*Code lines:* 57  
*Contained by:* module `dark_matter_profiles_einasto`  
*Modules used:* `numerical_constants_astronomical`  
`numerical_constants_physical`  
**function:** `dark_matter_profile_freefall_radius_increase_rate_einasto`  
*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:* module `dark_matter_profiles_einasto`  
*Modules used:* `numerical_constants_astronomical`  
`numerical_constants_physical`  
**function:** `dark_matter_profile_kspace_einasto`  
*Description:* Returns the Fourier transform of the Einasto density profile at the specified `waveNumber` (given in Mpc\(^{-1}\)).  
*Code lines:* 45  
*Contained by:* module `dark_matter_profiles_einasto`  
*Modules used:* `dark_matter_halo_scales`  
**function:** `dark_matter_profile_potential_einasto`  
*Description:* Returns the potential (in (km/s)\(^2\)) in the dark matter profile of `thisNode` at the given `radius` (given in units of Mpc).  
*Code lines:* 22  
*Contained by:* module `dark_matter_profiles_einasto`  
*Modules used:* `dark_matter_halo_scales`  
`numerical_constants_physical`  
**function:** `dark_matter_profile_rotation_normalization_einasto`  
*Description:* Return the rotation normalization of an Einasto halo density profile.  
*Code lines:* 22  
*Contained by:* module `dark_matter_profiles_einasto`
18.1. Program units

Modules used:
- dark_matter_halo_scales
- gamma_functions
- numerical_constants_math

subroutine: dark_matter_profiles_einasto_state_retrieve
Description: Retrieve the tabulation state from the file.
Code lines: 14
Contained by: module dark_matter_profiles_einasto

subroutine: dark_matter_profiles_einasto_state_store
Description: Write the tabulation state to file.
Code lines: 9
Contained by: module dark_matter_profiles_einasto

function: density_einasto_scale_free
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: 12
Contained by: module dark_matter_profiles_einasto
Modules used:
- gamma_functions
- numerical_constants_math

function: enclosed_mass_einasto_scale_free
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: 13
Contained by: module dark_matter_profiles_einasto
Modules used:
- gamma_functions

subroutine: energy_table_make
Description: Create a tabulation of the energy of Einasto profiles as a function of their concentration of alpha parameter.
Code lines: 96
Contained by: module dark_matter_profiles_einasto
Modules used:
- iso_c_binding
- memory_management
- numerical_constants_math
- numerical_integration
- numerical_interpolation
- numerical_ranges

function: fourier_profile_integrand_einasto
Description: Integrand for Einasto Fourier profile.
Code lines: 11
Contained by: module dark_matter_profiles_einasto
Modules used:
- iso_c_binding
- numerical_constants_math

subroutine: fourier_profile_table_make
Description: Create a tabulation of the Fourier transform of Einasto profiles as a function of their alpha parameter and dimensionless wavenumber.
Code lines: 102
Contained by: module dark_matter_profiles_einasto
Modules used:
- galacticus_display
- iso_c_binding
- memory_management
- numerical_integration
18. Source Code Documentation

```
numerical_interpolation numerical_ranges

function: freefall_time_scale_free
Description: Compute the freefall time in a scale-free Einasto halo.
Code lines: 16
Contained by: module dark_matter_profiles_einasto
Modules used: iso_c_binding numerical_integration

function: freefall_time_scale_free_integrand_einasto
Description: Integrand function used for finding the free-fall time in Einasto halos.
Code lines: 10
Contained by: module dark_matter_profiles_einasto
Modules used: iso_c_binding

function: jeans_equation_integrand_einasto
Description: Integrand for Einasto profile Jeans equation.
Code lines: 10
Contained by: module dark_matter_profiles_einasto
Modules used: iso_c_binding

function: kinetic_energy_integrand_einasto
Description: Integrand for Einasto profile kinetic energy.
Code lines: 10
Contained by: module dark_matter_profiles_einasto
Modules used: iso_c_binding

function: potential_einasto_scale_free
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: 13
Contained by: module dark_matter_profiles_einasto
Modules used: gamma_functions

function: potential_energy_integrand_einasto
Description: Integrand for Einasto profile potential energy.
Code lines: 10
Contained by: module dark_matter_profiles_einasto
Modules used: iso_c_binding

function: radius_from_specific_angular_momentum_einasto
Description: Returns the radius (in Mpc) in thisNode 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_einasto
Modules used: numerical_constants_physical

function: radius_from_specific_angular_momentum_scale_free
Description: Compute the radius at which a circular orbit has the given specificAngularMomentumScaleFree in a scale free Einasto profile.
```
18.1. Program units

Code lines: 33
Contained by: module dark_matter_profiles_einasto
Modules used: numerical_interpolation

subroutine: radius_from_specific-angular-momentum-table-make
Description: Create a tabulation of the relation between specific angular momentum and radius in an Einasto profile.
Code lines: 70
Contained by: module dark_matter_profiles_einasto
Modules used: gamma_functions numerical_interpolation memory_management numerical_ranges

file: dark_matter_profiles.F90
Description: Contains a module which implements calculations related to the dark matter halo density profile.
Code lines: 290

module: dark_matter_profiles
Description: Implements calculations related to the dark matter halo density profile.
Code lines: 270
Contained by: file dark_matter_profiles.F90
Modules used: galacticus_nodes iso_varying_string
Used by: function freefall_radius_dark_matter-halo function cooling_specific-angular-momentum-constant-rotation function dark_matter_halo_angular-momentum_growth_rate subroutine solve_for_radius function galactic_structure_radius-initial_adiabatic_solver subroutine galacticus_extra_output-halo-fourier-profile function half_mass_radius_root function satellite_time_until_merging-boylankolchin2008 function satellite_time_until_merging-jiang2008 function pericenter_solver

function: dark_matter_profile_circular-velocity
Description: Returns the circular velocity (in km/s) in the dark matter profile of thisNode at the given radius (given in units of Mpc).
Code lines: 14
Contained by: module dark_matter_profiles

function: dark_matter_profile_density
Description: Returns the density (in $M_\odot \text{Mpc}^{-3}$) in the dark matter profile of thisNode at the given radius (given in units of Mpc).
Code lines: 14
function: dark_matter_profile_enclosed_mass
Description: Returns the enclosed mass (in $M_\odot$) in the dark matter profile of thisNode at the given radius (given in units of Mpc).
Code lines: 14
Contained by: module dark_matter_profiles

function: dark_matter_profile_energy
Description: Returns the total energy of thisNode in units of $M_\odot$ km$^2$ s$^{-1}$.
Code lines: 12
Contained by: module dark_matter_profiles

function: dark_matter_profile_energy_growth_rate
Description: Returns the rate of change of the total energy of thisNode in units of $M_\odot$ km$^2$ s$^{-1}$ Gyr$^{-1}$.
Code lines: 12
Contained by: module dark_matter_profiles

function: dark_matter_profile_freefall_radius
Description: Returns the freefall radius (in Mpc) corresponding to the given time (in Gyr) in thisNode.
Code lines: 13
Contained by: module dark_matter_profiles

function: dark_matter_profile_freefall_radius_increase_rate
Description: Returns the rate of increase of the freefall radius (in Mpc/Gyr) corresponding to the given time (in Gyr) in thisNode.
Code lines: 14
Contained by: module dark_matter_profiles

subroutine: dark_matter_profile_initialize
Description: Initialize the dark matter profile module.
Code lines: 56
Contained by: module dark_matter_profiles

Modules used:
- dark_matter_profiles_einasto
- dark_matter_profiles_isothermal
- dark_matter_profiles_isothermal
- dark_matter_profiles_nfw
- galacticus_error
- input_parameters

function: dark_matter_profile_kspace
Description: Returns the normalized Fourier space density profile of the dark matter profile of thisNode at the given waveNumber (given in units of Mpc$^{-1}$).
Code lines: 14
Contained by: module dark_matter_profiles

function: dark_matter_profile_potential
Description: Returns the gravitational potential (in (km/s)$^2$) in the dark matter profile of thisNode at the given radius (given in units of Mpc).
Code lines: 14
Contained by: module dark_matter_profiles

function: dark_matter_profile_radius_from_specific_angular_momentum
18.1. Program units

**Description:** Returns the radius (in Mpc) in the dark matter profile of thisNode at which the specific angular momentum of a circular orbit equals specificAngularMomentum (specified in units of km s\(^{-1}\) Mpc.

**Code lines:** 14

**Contained by:** module dark_matter_profiles

**function: dark_matter_profile_rotation_normalization**

**Description:** Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for thisNode. Specifically, the normalization, \(A\), returned is such that \(V_{\text{rot}} = AJ/M\).

**Code lines:** 13

**Contained by:** module dark_matter_profiles

**file: dark_matter_profiles.NFW.F90**

**Description:** Contains a module which implements NFW halo profiles.

**Code lines:** 812

**module: dark_matter_profiles_nfw**

**Description:** Implements NFW halo profiles.

**Code lines:** 792

**Contained by:** file dark_matter_profiles.NFW.F90

**Modules used:** f gsl

**kind_numbers**

**tables**

**Used by:**

subroutine dark_matter_profile_-initialize

subroutine galacticus_calculations_-reset

subroutine galacticus_state_retrieve

subroutine galacticus_state_store

**function: angular_momentum_nfw_scale_free**

**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:

\[
J = \int_0^c 4\pi x^3 \rho(x)dx \left/ \int_0^c 4\pi x^2 \rho(x)dx \right.,
\]

where \(x\) is radius in units of the scale radius and \(c\) is concentration. This can be evaluated to give

\[
J = \left[1 + c - 2 \ln(1 + c) - \frac{1}{1 + c}\right] \left/ \ln(1 + c) - \frac{c}{1 + c}\right.,
\]

(18.3)

(18.4)

**Code lines:** 15

**Contained by:** module dark_matter_profiles_nfw

**function: dark_matter_profile_circular_velocity_nfw**

**Description:** Returns the circular velocity (in km/s) in the dark matter profile of thisNode at the given radius (given in units of Mpc). For an NFW halo this is independent of radius and therefore equal to the virial velocity.

**Code lines:** 14

**Contained by:** module dark_matter_profiles_nfw

**Modules used:** numerical_constants_physical

**function: dark_matter_profile_density_nfw**
18. Source Code Documentation

Description: Returns the density (in $M_\odot\,\text{Mpc}^{-3}$) in the dark matter profile of thisNode at the given radius (given in units of Mpc).

Code lines: 18

Contained by: module dark_matter_profiles_nfw

Modules used: dark_matter_halo_scales

function: dark_matter_profile_enclosed_mass_nfw

Description: Returns the enclosed mass (in $M_\odot$) in the dark matter profile of thisNode at the given radius (given in units of Mpc).

Code lines: 18

Contained by: module dark_matter_profiles_nfw

Modules used: dark_matter_halo_scales

function: dark_matter_profile_energy_growth_rate_nfw

Description: Returns the rate of change of the energy of an NFW halo density profile.

Code lines: 28

Contained by: module dark_matter_profiles_nfw

Modules used: dark_matter_halo_scales

function: dark_matter_profile_energy_nfw

Description: Return the energy of an NFW halo density profile.

Code lines: 24

Contained by: module dark_matter_profiles_nfw

Modules used: dark_matter_halo_scales

function: dark_matter_profile_freefall_radius_increase_rate_nfw

Description: Returns the rate of increase of the freefall radius in the NFW density profile at the specified time (given in Gyr).

Code lines: 43

Contained by: module dark_matter_profiles_nfw

Modules used: dark_matter_halo_scales numerical_constants_astronomical

function: dark_matter_profile_freefall_radius_nfw

Description: Returns the freefall radius in the NFW density profile at the specified time (given in Gyr).

Code lines: 42

Contained by: module dark_matter_profiles_nfw

Modules used: dark_matter_halo_scales numerical_constants_astronomical

function: dark_matter_profile_kspace_nfw

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: 26

Contained by: module dark_matter_profiles_nfw

Modules used: dark_matter_halo_scales exponential_integrals

subroutine: dark_matter_profile_nfw_freefall_tabulate

Description: Tabulates the freefall time vs. freefall radius for NFW halos.

Code lines: 41

Contained by: module dark_matter_profiles_nfw
18.1. Program units

**subroutine:** dark_matter_profile_nfw_initialize

*Description:* Initializes the “NFW” halo profile module.
*Code lines:* 38
*Contained by:* module dark_matter_profiles_nfw
*Modules used:* galacticus_error iso_varying_string

**subroutine:** dark_matter_profile_nfw_inverse_angular_momentum

*Description:* Tabulates the specific angular momentum vs. radius in an NFW profile for rapid inversion.
*Code lines:* 43
*Contained by:* module dark_matter_profiles_nfw

**subroutine:** dark_matter_profile_nfw_reset

*Description:* Reset the cooling radius calculation.
*Code lines:* 8
*Contained by:* module dark_matter_profiles_nfw

**subroutine:** dark_matter_profile_nfw_tabulate

*Description:* Tabulate properties of the NFW halo profile which must be computed numerically.
*Code lines:* 36
*Contained by:* module dark_matter_profiles_nfw

**function:** dark_matter_profile_potential_nfw

*Description:* Returns the potential \((\text{in} \ (\text{km/s})^2)\) in the dark matter profile of \texttt{thisNode} at the given radius \((\text{given in units of Mpc})\).
*Code lines:* 23
*Contained by:* module dark_matter_profiles_nfw
*Modules used:* dark_matter_halo_scales

**function:** dark_matter_profile_rotation_normalization_nfw

*Description:* Return the normalization of the rotation velocity vs. specific angular momentum relation.
*Code lines:* 22
*Contained by:* module dark_matter_profiles_nfw
*Modules used:* dark_matter_halo_scales

**subroutine:** dark_matter_profiles_nfw_state_retrieve

*Description:* Retrieve the tabulation state from the file.
*Code lines:* 15
*Contained by:* module dark_matter_profiles_nfw

**subroutine:** dark_matter_profiles_nfw_state_store

*Description:* Write the tabulation state to file.
*Code lines:* 8
*Contained by:* module dark_matter_profiles_nfw

**function:** density_nfw_scale_free
18. Source Code Documentation

**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:** 9

**Contained by:** module `dark_matter_profiles_nfw`
18.1. Program units

Modules used: numerical_constants_math

function: enclosed_mass_nfw_scale_free
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_nfw

function: freefall_time_scale_free
Description: Compute the freefall time in a scale-free NFW halo.
Code lines: 24
Contained by: module dark_matter_profiles_nfw
Modules used: iso_c_binding numerical_integration

function: freefall_time_scale_free_integrand
Description: Integrand function used for finding the free-fall time in NFW halos.
Code lines: 24
Contained by: module dark_matter_profiles_nfw
Modules used: iso_c_binding

function: jeans_equation_integrand
Description: Integrand for NFW profile Jeans equation.
Code lines: 10
Contained by: module dark_matter_profiles_nfw
Modules used: iso_c_binding

function: kinetic_energy_integrand
Description: Integrand for NFW profile kinetic energy.
Code lines: 10
Contained by: module dark_matter_profiles_nfw
Modules used: iso_c_binding

function: nfw_profile_energy
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: 40
Contained by: module dark_matter_profiles_nfw
Modules used: iso_c_binding numerical_constants_math numerical_integration

function: potential_energy_integrand
Description: Integrand for NFW profile potential energy.
Code lines: 10
Contained by: module dark_matter_profiles_nfw
Modules used: iso_c_binding

function: radius_from_specific_angular_momentum_nfw
Description: Returns the radius (in Mpc) in thisNode 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: 50
18.1. Program units

contained by: module dark_matter_profiles_nfw

function: specific_angular_momentum_nfw_scale_free
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: 8
Contained by: module dark_matter_profiles_nfw

file: dark_matter_profiles.isothermal.F90
Description: Contains a module which implements a isothermal halo spin distribution.
Code lines: 232

module: dark_matter_profiles_isothermal
Description: Implements a isothermal halo spin distribution.
Code lines: 212
Contained by: file dark_matter_profiles.isothermal.F90
Used by: subroutine dark_matter_profile_initialize

function: dark_matter_profile_circular_velocity_isothermal
Description: Returns the circular velocity (in km/s) in the dark matter profile of thisNode 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: 11
Contained by: module dark_matter_profiles_isothermal
Modules used: dark_matter_halo_scales galacticus_nodes

function: dark_matter_profile_density_isothermal
Description: Returns the density (in $M_\odot$ Mpc$^{-3}$) in the dark matter profile of thisNode at the given radius (given in units of Mpc).
Code lines: 14
Contained by: module dark_matter_profiles_isothermal
Modules used: dark_matter_halo_scales galacticus_nodes numerical_constants_math

function: dark_matter_profile_enclosed_mass_isothermal
Description: Returns the enclosed mass (in $M_\odot$) in the dark matter profile of thisNode at the given radius (given in units of Mpc).
Code lines: 13
Contained by: module dark_matter_profiles_isothermal
Modules used: dark_matter_halo_scales galacticus_nodes

function: dark_matter_profile_energy_growth_rate_isothermal
Description: Return the rate of change of the energy of an isothermal halo density profile.
Code lines: 11
Contained by: module dark_matter_profiles_isothermal
Modules used: dark_matter_halo_scales galacticus_nodes

function: dark_matter_profile_energy_isothermal
Source Code Documentation

**Description:** Return the energy of an isothermal halo density profile.

**Code lines:** 11  
**Contained by:** module `dark_matter_profiles_isothermal`  
**Modules used:** `dark_matter_halo_scales` `galacticus_nodes`

**Function:** `dark_matter_profile_freefall_radius_increase_rate_isothermal`  
**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:

\[
\dot{r}_{ff}(t) = \sqrt{\frac{2}{\pi}} V_{\text{virial}}. 
\]

(18.5)

**Code lines:** 15  
**Contained by:** module `dark_matter_profiles_isothermal`  
**Modules used:** `dark_matter_halo_scales` `galacticus_nodes` `numerical_constants_astronomical`

**Function:** `dark_matter_profile_freefall_radius_isothermal`  
**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:

\[
r_{ff}(t) = \sqrt{\frac{2}{\pi}} V_{\text{virial}} t. 
\]

(18.6)

**Code lines:** 15  
**Contained by:** module `dark_matter_profiles_isothermal`  
**Modules used:** `dark_matter_halo_scales` `galacticus_nodes` `numerical_constants_astronomical`

**Subroutine:** `dark_matter_profile_isothermal_initialize`  
**Description:** Initializes the “Isothermal” halo spin distribution module.

**Code lines:** 31  
**Contained by:** module `dark_matter_profiles_isothermal`  
**Modules used:** `iso_varying_string`

**Function:** `dark_matter_profile_kspace_isothermal`  
**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:** 21  
**Contained by:** module `dark_matter_profiles_isothermal`  
**Modules used:** `dark_matter_halo_scales` `exponential_integrals` `galacticus_nodes`

**Function:** `dark_matter_profile_potential_isothermal`  
**Description:** Returns the potential (in \((\text{km/s})^2\)) in the dark matter profile of thisNode at the given radius (given in units of Mpc).

**Code lines:** 13  
**Contained by:** module `dark_matter_profiles_isothermal`  
**Modules used:** `dark_matter_halo_scales` `galacticus_error` `galacticus_nodes`
function: dark_matter_profile_rotation_normalization_isothermal
Description: Return the normalization of the rotation velocity vs. specific angular momentum relation.
Code lines: 9
Contained by: module dark_matter_profiles_isothermal
Modules used: dark_matter_halo_scales galacticus_nodes

function: radius_from_specific-angular_momentum_isothermal
Description: Returns the radius (in Mpc) in thisNode 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, \( r = j/V_{\text{virial}} \) where \( j (= \text{specificAngularMomentum}) \) is the specific angular momentum and \( r \) the required radius.
Code lines: 13
Contained by: module dark_matter_profiles_isothermal
Modules used: dark_matter_halo_scales galacticus_nodes

file: dark_matter_profiles.structure.concentration.F90
Description: Contains a module which implements calculations of dark matter halo density profile concentrations.
Code lines: 121

module: dark_matter_profiles_concentrations
Description: Implements calculations of dark matter halo density profile concentrations.
Code lines: 101
Contained by: file dark_matter_profiles.structure.concentration.F90
Modules used: galacticus_nodes iso_varying_string
Used by: subroutine node_component_dark_matter_-profile_scale_initialize_scale program test_nfw96_concentration_-profile_scale_initialize_scale dark_energy program test_prada2011_concentration program test_zhao2009_dark_energy program test_zhao2009_open

subroutine: dark_matter_concentrations_initialize
Description: Initialize the dark matter profile module.
Code lines: 60
Contained by: module dark_matter_profiles_concentrations

function: dark_matter_profile_concentration
Description: Returns the concentration of the dark matter profile of thisNode.
Code lines: 12
Contained by: module dark_matter_profiles_concentrations
18. Source Code Documentation

Description: Contains a module which implements the Gao et al. [2008] NFW halo concentration algorithm.
Code lines: 67

module: dark_matter_profiles_concentrations_gao2008
Description: Implements the Gao et al. [2008] NFW halo concentration algorithm.
Code lines: 47
Used by: subroutine dark_matter_concentrations_gao2008_initialize

subroutine: dark_matter_concentrations_gao2008_initialize
Description: Initializes the “Gao2008” halo concentration module.
Code lines: 9
Contained by: module dark_matter_profiles_concentrations_gao2008
Modules used: iso_varying_string
cosmology_functions
galacticus_nodes

function: dark_matter_profile_concentration_gao2008
Description: Returns the concentration of the dark matter profile of thisNode using the method of Gao et al. [2008]. More specifically, we fit the redshift dependence of the parameters $A$ and $B$ in the fitting formula of Gao et al. [2008] and use these fits to find $A$ and $B$ at any given redshift, from which we then compute the concentration. Note that the fits of Gao et al. [2008] were computed using Einasto profile fits and utilizing $M_{200}$ and $c_{200}$.
Code lines: 23
Contained by: module dark_matter_profiles_concentrations_gao2008
Modules used: cosmology_functions
galacticus_nodes

Description: Contains a module which implements the Muñoz-Cuartas et al. [2011] NFW halo concentration algorithm.
Code lines: 66

module: dark_matter_profiles_concentrations_munozcuartas2011
Description: Implements the Muñoz-Cuartas et al. [2011] NFW halo concentration algorithm.
Code lines: 46
Used by: subroutine dark_matter_concentrations_munozcuartas2011_initialize

subroutine: dark_matter_concentrations_munozcuartas2011_initialize
Description: Initializes the “Munoz-Cuartas2011” halo concentration module.
Code lines: 10
Contained by: module dark_matter_profiles_concentrations_munozcuartas2011
Modules used: iso_varying_string

function: dark_matter_profile_concentration_munozcuartas2011
Description: Returns the concentration of the dark matter profile of thisNode using the method of Muñoz-Cuartas et al. [2011].
18.1. Program units

Code lines: 21
Contained by: module dark_matter_profiles_concentrations_munozcuartas2011
Modules used: cosmological_parameters cosmology_functions galacticus_nodes

file: dark_matter_profiles.structure.concentration.NFW.F90
Description: Contains a module which implements the Navarro et al. [1996] NFW halo concentration algorithm.
Code lines: 128

module: dark_matter_profiles_concentrations_nfw1996
Description: Implements the Navarro et al. [1996] NFW halo concentration algorithm.
Code lines: 108
Contained by: file dark_matter_profiles.structure.concentration.NFW.F90
Used by: subroutine dark_matter_concentrations_nfw1996_initialize

subroutine: dark_matter_concentrations_nfw1996_initialize
Description: Initializes the “NFW1996” halo concentration module.
Code lines: 36
Contained by: module dark_matter_profiles_concentrations_nfw1996
Modules used: input_parameters iso_varying_string

function: dark_matter_profile_concentration_nfw1996
Description: Returns the concentration of the dark matter profile of thisNode using the method of Navarro et al. [1996].
Code lines: 42
Contained by: module dark_matter_profiles_concentrations_nfw1996
Modules used: cosmology_functions critical_overdensity galacticus_nodes power_spectra root_finder virial_density_contrast

function: nfw_concentration_function_root
Description: Root function used in finding concentrations in the Navarro et al. [1996] method.
Code lines: 7
Contained by: module dark_matter_profiles_concentrations_nfw1996

Description: Contains a module which implements the Prada et al. [2011] halo concentration algorithm.
Code lines: 240

module: dark_matter_profiles_concentrations_prada2011
Description: Implements the Prada et al. [2011] halo concentration algorithm.
Code lines: 220
Contained by: file dark_matter_profiles.structure.concentration.Prada2011.F90
Used by: subroutine dark_matter_concentrations_prada2011_initialize

function: b0
18. Source Code Documentation

*Function: b₀*

**Description:** The function $B_0(x)$ as defined in eqn. (18) of Prada et al. [2011].

**Code lines:** 5

**Contained by:** module `dark_matter_profiles_concentrations_prada2011`

*Function: b₁*

**Description:** The function $B_1(x)$ as defined in eqn. (18) of Prada et al. [2011].

**Code lines:** 5

**Contained by:** module `dark_matter_profiles_concentrations_prada2011`

*Function: c*

**Description:** The function $C(\sigma')$ as defined in eqn. (17) of Prada et al. [2011].

**Code lines:** 5

**Contained by:** module `dark_matter_profiles_concentrations_prada2011`

*Function: cₘᵢₙᵢₙ*

**Description:** The function $c_{\min}(x)$ as defined in eqn. (19) of Prada et al. [2011].

**Code lines:** 6

**Contained by:** module `dark_matter_profiles_concentrations_prada2011`

**Modules used:** `numerical_constants_math`

*Subroutine: dark_matter_concentrations_prada2011_initialize*

**Description:** Initializes the “Prada2011” halo concentration module.

**Code lines:** 146

**Contained by:** module `dark_matter_profiles_concentrations_prada2011`

**Modules used:** `input_parameters iso_varying_string`

*Function: dark_matter_profile_concentration_prada2011*

**Description:** Returns the concentration of the dark matter profile of thisNode using the method of Prada et al. [2011].

**Code lines:** 19

**Contained by:** module `dark_matter_profiles_concentrations_prada2011`

**Modules used:** `cosmological_parameters cosmology_functions galacticus_nodes linear_growth power_spectra`

*Function: inversesigmamin*

**Description:** The function $\sigma_{\min}^{-1}(x)$ as defined in eqn. (20) of Prada et al. [2011].

**Code lines:** 6

**Contained by:** module `dark_matter_profiles_concentrations_prada2011`

**Modules used:** `numerical_constants_math`

*File: dark_matter_profiles.structure.concentration.Zhao2009.F90*

**Description:** Contains a module which implements the Zhao et al. [2009] NFW halo concentration algorithm.

**Code lines:** 67

*Module: dark_matter_profiles_concentrations_zhao2009*

**Description:** Implements the Zhao et al. [2009] NFW halo concentration algorithm.

**Code lines:** 47

**Contained by:** file `dark_matter_profiles.structure.concentration.Zhao2009.F90`
18.1. Program units

**Used by:** subroutine `dark_matter_concentrations_initialize`

**subroutine:** `dark_matter_concentrations_zhao2009_initialize`

*Description:* Initializes the “Zhao2009” halo concentration module.

*Code lines:* 10

*Contained by:* module `dark_matter_profiles_concentrations_zhao2009`

*Modules used:* `iso_varying_string`

**function:** `dark_matter_profile_concentration_zhao2009`

*Description:* Returns the concentration of the dark matter profile of `thisNode` using the method of Zhao et al. [2009].

*Code lines:* 22

*Contained by:* module `dark_matter_profiles_concentrations_zhao2009`

*Modules used:* `dark_matter_halo_formation_times`, `galacticus_nodes`

**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`.

*Code lines:* 12

*Contained by:* module `dark_matter_profiles_shapes`

**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`, `input_parameters`

**file:** `dark_matter_profiles.structure.shape.Gao2008.F90`

*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_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:

\[
\alpha = \begin{cases} 
0.155 + 0.0095\nu^2 & \text{if } \nu < 3.907 \\
0.3 & \text{if } \nu \geq 3.907, 
\end{cases}
\]  

(18.7)

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: 203

module: dark_matter_profile_structure_tasks

Description: Implements structure tasks related to the dark matter halo density profile.

Code lines: 183

Contained by: file dark_matter_profiles.structure_tasks.F90

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: 39

Contained by: module dark_matter_profile_structure_tasks

Modules used: cosmological_parameters galacticus_nodes 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: 46

Contained by: module dark_matter_profile_structure_tasks

Modules used: cosmological_parameters galacticus_nodes galactic_structure_options
**galactic_structure_options**

**function: dark_matter_profile_potential_task**
*Description:* Return the potential due to dark matter.
*Code lines:* 20
*Contained by:* module dark_matter_profile_structure_tasks
*Modules used:* galactic_structure_options
galacticus_error

**function: dark_matter_profile_rotation_curve_gradient_task**
*Description:* Computes the rotation curve gradient for the dark matter.
*Code lines:* 24
*Contained by:* module dark_matter_profile_structure_tasks
*Modules used:* galactic_structure_options
numerical_constants_math
numerical_constants_physical

**function: dark_matter_profile_rotation_curve_task**
*Description:* Computes the rotation curve at a given radius for a dark matter profile.
*Code lines:* 20
*Contained by:* module dark_matter_profile_structure_tasks
*Modules used:* galactic_structure_options
numerical_constants_physical

**file: events.branch_jump.F90**
*Description:* Contains a module which handles node branch jump events.
*Code lines:* 74

**module: node_branch_jumps**
*Description:* Handles satellite node branch jump events.
*Code lines:* 54
*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
galacticus_nodes
iso_varying_string
merger_trees_evolve_deadlock_status
string_handling

**file: events.halo_formation.F90**
*Description:* Contains a module which performs tasks associated with “halo formation” events.
*Code lines:* 66

**module: events_halo_formation**
*Description:* Performs tasks associated with “halo formation” events.
*Code lines:* 46
*Contained by:* file events.halo_formation.F90
*Used by:* subroutine node_component_formation_-
time_cole2000_create
**subroutine: event_halo_formation**
*Description:* Perform tasks associated with a “halo formation” event in `thisNode`.
*Code lines:* 36
*Contained by:* module `events_halo_formation`
*Modules used:* `galacticus_nodes` `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` `satellite_promotion`
`string_handling`

**subroutine: events_node_merger_initialize_slh**
*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`
18.1. Program units

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: 138

module: galactic_dynamics_bar_instabilities_eln
Description: Implements calculations of bar instability based on the Efstathiou et al. [1982] criterion.
Code lines: 118
Contained by: file galactic_dynamics.bar_instability.ELN.F90
Used by: subroutine galactic_dynamics_bar_instability_initialize

function: 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: 64
Contained by: module galactic_dynamics_bar_instabilities_eln
Modules used: galacticus_nodes numerical_constants_astronomical 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 iso_varying_string

file: galactic_dynamics.bar_instability.F90
Description: Contains a module which implements calculations of bar instability in galactic disks.
Code lines: 116
module: galactic_dynamics_bar_instabilities
Description: Implements calculations of bar instability in galactic disks.
Code lines: 96
Contained by: file galactic_dynamics.bar_instability.F90
Modules used: galacticus_nodes iso_varying_string
Used by: subroutine node_component_disk_exponential_rate_compute

function: 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.
Code lines: 13
Contained by: module galactic_dynamics_bar_instabilities

subroutine: galactic_dynamics_bar_instability_initialize
Description: Initialize the bar instability module.
Code lines: 54
Contained by: module galactic_dynamics_bar_instabilities
Modules used: galactic_dynamics_bar_instabilities_eln galactic_dynamics_bar_instabilities_null
galacticus_error input_parameters

file: galactic_dynamics.bar_instability.null.F90
Description: Contains a module which implements a null calculation of bar instability.
Code lines: 55

module: galactic_dynamics_bar_instabilities_null
Description: Implements a null calculation of bar instability.
Code lines: 35
Contained by: file galactic_dynamics.bar_instability.null.F90
Modules used: galacticus_nodes
Used by: subroutine galactic_dynamics_bar_instability_initialize

function: bar_instability_timescale_null
Description: Assumes that disks are never bar unstable and so returns an infinite timescale for bar instability.
Code lines: 9
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
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.
18.1. Program units

**module: galactic_structure_densities**
- **Description:** Implements calculations of the density at a specific position.
- **Code lines:** 119
- **Contained by:** file galactic_structure.density.F90
- **Used by:**
  - function black_hole_binary_separationgrowth_rate_standard
  - function velocity_dispersion_integrand
  - function galacticus_output_trees-
    - vlcty Dispersion vlcty dnsty_intgrnd
  - function galacticus_output_trees-
    - vlcty dsprsn vlcty dnsty srfc_intgrnd

**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:** 94
- **Contained by:** module galactic_structure_densities
- **Modules used:**
  - coordinate_systems
  - dark_matter_profile_structure_tasks
  - galactic_structure_options
  - galacticus_error
  - galacticus_nodes
  - hot_halo_density_profile

**file: galactic_structure.enclosed_mass.F90**
- **Description:** Contains a module which implements calculations of the mass enclosed within a specified radius.
- **Code lines:** 204

**module: galactic_structure_enclosed_masses**
- **Description:** Implements calculations of the mass enclosed within a specified radius.
- **Code lines:** 184
- **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
  - function galactic_structure_velocity-
    - dispersion
  - subroutine galacticus_output_tree-
    - density contrast
  - subroutine galacticus_merge_tree-
    - output_filter_luminosity
  - function mean_density_contrast_root
  - subroutine galacticus_merge_tree-
    - output_filter_stellar_mass
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: 52
   Contained by: module galactic_structure_enclosed_masses
   Modules used: dark_matter_profile_structure_tasks hot_halo_density_profile

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: 56
   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.
module: galactic_structure_options
Description: Provides various internal option codes for the galactic structure functions.
Code lines: 115
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_enclosed_mass_task
function dark_matter_profile_rotation_curve_gradient_task
function galactic_structure_density

module galactic_structure_potentials
module galactic_structure_initial_radii_adiabatic
module galactic_structure_rotation_curve_gradients
function galactic_structure_velocity_dispersion
subroutine galacticus_output_tree_density_contrast_initialize
subroutine galacticus_merger_tree_output_filter_luminosity
subroutine galacticus_output_tree_half_light
subroutine galacticus_output_tree_rotation_curve
subroutine galacticus_output_tree_velocity_dispersion
function hot_halo_profile_density_task
function hot_halo_profile_rotation_curve_gradient_task
function hot_halo_ram_pressure_stripping_radius_solver
subroutine merger_tree_record_evolution_store
function node_component_black_hole_simple_potential
function node_component_black_hole_simple_rotation_curve
subroutine node_component_black_hole_standard_mass_accretion_rate

function tidal_radius_root
function dark_matter_profile_density_task
function dark_matter_profile_potential_task
function dark_matter_profile_rotation_curve_task
module galactic_structure_enclosed_masses
subroutine solve_for_radius
module galactic_structure_rotation_curves
module galactic_structure_surface_densities
subroutine galacticus_output_tree_density_contrast
function mean_density_contrast_root
subroutine galacticus_merger_tree_output_filter_stellar_mass
subroutine galacticus_output_tree_mass_profile
subroutine galacticus_output_tree_rotation_curve_initialize
subroutine galacticus_output_tree_velocity_dispersion_initialize
function hot_halo_profile_enclosed_mass_task
function hot_halo_profile_rotation_curve_task
subroutine merger_tree_history_store
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_satellite_merging
function node_component_black_hole_standard_potential
function node_component_black_hole_standard_rotation_curve
subroutine node_component_black_hole_standard_node_merger
function ram_pressure_stripping_mass_loss_rate_disk_simple
subroutine satellite_merging_mass_movement_simple
function half_mass_radius_root_cole2000
subroutine satellite_merging_remnant_progenitor_properties_standard
function star_formation_feedback_disk_outflow_rate_creonsey2012_integrand
function star_formation_rate_surface_density_disk_br
function star_formation_rate_surface_density_disk_ks
function star_formation_rate_surface_density_disk_exschmidt

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
galacticus_nodes
Modules used: galactic_structure_options

Used by: subroutine node_component_black_hole_standard_merge_black_holes
galacticus_nodes
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

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
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: 299

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: 278
Contained by: file galactic_structure.radius_solver.adiabatic.F90
18. Source Code Documentation

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: 97
Contained by: module galactic_structure_radii_adiabatic
Modules used: cosmological_parameters 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: 105
Contained by: module galactic_structure_radii_adiabatic
Modules used: dark_matter_profiles galacticus_error
galactic_structure_initial_radii galactic_structure_rotation_curves
galacticus_error iso_varying_string
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: 149

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: 128
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 iso_varying_string
18.1. Program units

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: 24
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: galactic_structure_initial_radii_adiabatic
galactic_structure_initial_radii_static

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_initial_initialize
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: 413

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: 392
Contained by: file galactic_structure.radius_solver.initial_radii.adiabatic.F90
Modules used: galactic_structure_options galacticus_nodes
Used by: subroutine galactic_structure_radius_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
Description: Returns the derivative of 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: 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
18.1. Program units

**subroutine:** galactic_structure_radii_initial_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:* 163  
*Contained by:* module galactic_structure_initial_radii_adiabatic  
*Modules used:* cosmological_parameters, dark_matter_halo_scales, hot_halo_density_profile, node_component_black_hole_simple_structure, node_component_black_hole_standard_structure, numerical_constants_physical, 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  
*Description:* Root function used in finding the initial radius in the dark matter halo when solving for adiabatic contraction.  
*Code lines:* 14  
*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:* 18  
*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
18. Source Code Documentation

Modules used: galacticus_nodes
Used by: subroutine galactic_structure_radius_initial_initialize

subroutine: galactic_structure_initial_radii_static_initialize
Description: Initializes 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: 127

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: 106
Contained by: file galactic_structure.radius_solver.linear.F90
Modules used: galactic_structure.radius_solver_procedures
Used by: subroutine galactic_structure_radii_solve_linear

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_scale
node_component_spheroid_standard

subroutine: solve_for_radius
Description: Solve for the equilibrium radius of the given component.
Code lines: 21
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

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: 169

module: galactic_structure_radii_simple
Description: Implements a simple galactic radii solver (no adiabatic contraction and no self-gravity of baryons).
Code lines: 149
Contained by: file galactic_structure.radius_solver.simple.F90
Modules used: galactic_structure_radius_solver_procedures
Used by: subroutine galactic_structure_radii_solve_simple

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
subroutine: solve_for_radius
Description: Solve for the equilibrium radius of the given component.
Code lines: 23
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: 116

module: galactic_structure_rotation_curves
Description: Implements calculations of the rotation curve as a specified radius.
Code lines: 96
Contained by: file galactic_structure.rotation_curve.F90
Modules used: galactic_structure_options galacticus_nodes
Used by: function black_hole_binary_separation_growth_rate_standard subroutine solve_for_radius
galactic_structure_rotation_curve gradient subroutine galacticus_output_tree_rotation_curve

function: component_rotation_curve
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: galactic_structure_rotation_curve
Description: Solve for the rotation curve at a given radius. Assumes that galacticus structure has already been solved for.
Code lines: 70
Contained by: module galactic_structure_rotation_curves
Modules used: dark_matter_profile_structure_tasks hot_halo_density_profile
node_component_black_hole_simple_structure node_component_black_hole_standard_structure
galactic_structure_rotation_curve gradient subroutine galacticus_output_tree_rotation_curve

file: galactic_structure.rotation_curve.gradient.F90
Description: Contains a module which implements calculations of the gradient of the rotation curve.
Code lines: 132

module: galactic_structure_rotation_curve_gradients
Description: Implements calculations of the rotation curve gradient
Code lines: 98
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: 73
Contained by: module `galactic_structure_rotation_curve_gradients`
Modules used:
- `dark_matter_profile_structure_tasks`
- `galactic_structure_rotation_curves`
- `hot_halo_density_profile`
- `node_component_black_hole_simple_structure`
- `node_component_black_hole_standard_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: 120

module: `galactic_structure_surface_densities`
Description: Implements calculations of the surface density at a specific position.
Code lines: 100
Contained by: file `galactic_structure.surface_density.F90`
Modules used:
- `galactic_structure_options`
- `galacticus_nodes`
Used by:
- function `ram_pressure_stripping_mass_loss_rate_disk_simple`
- function `star_formation_feedback_disk_outflow_rate_creasey2012_integrand`
- function `star_formation_rate_surface_density_disk_br`
- function `star_formation_rate_surface_density_disk_kmt09`
- function `star_formation_rate_surface_density_disk_ks`
- function `star_formation_rate_surface_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: 74
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.
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_grow_rate_standard
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: f gsl
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_density
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
18.1. Program units

**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:** 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 calculations for a new or updated node.

**Code lines:** 55

**Contained by:** module `galacticus_calculations_resets`

**Modules used:**
- `cooling_radii_isothermal`
- `cooling_radii_simple`
- `cooling_specific-angular_momenta-constant_rotation`
- `dark_matter_profiles_nfw`
- `node_component_disk_exponential`
- `node_component_hot_halo_standard`
- `node_component_hot_halo_very_simple`
- `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_d disks_ks`
- `star_formation_rate_surface_density_d disks_scaling`
- `dark_matter_halo_scales`

**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` function `black_hole_binary_separation_growth_rate_standard`

subroutine `chemical_state_cie_file_read` subroutine `cooling_function_cie_file_read`

subroutine `cooling_function_not_matched` function `h_0`

subroutine `dark_matter_profile_einstato_freefall_tabulate` subroutine `fourier_profile_table_make`

function `node_branch_jump` function `node_subhalo_promotion`

function `galactic_structure_radius_enclosing_mass` module `galacticus_tasks_basic`

function `galacticus_task_evolve_tree`

function `generalized_press_schechter_branch_mass` subroutine `halo_mass_function_compute`

function `generalized_press_schechter_subresolution_fraction`
subroutine merger_tree_construct_fully_specified
subroutine build_descendent_pointers
subroutine merger_tree_read_initialize
subroutine merger_tree_evolve_to
subroutine tree_node_odes_error_handler
subroutine evolve_to_time_report
subroutine abundances_dump
subroutine history_dump
subroutine merger_tree_data_structure_read_ascii
subroutine blackholecreatelinked
subroutine diskcreatelinked
subroutine hotalocreatelinked
subroutine interoutputcreatelinked
subroutine node_component_basic_dump
subroutine node_component_basicnull_dump
subroutine node_component_blackhole_dump
subroutine node_component_blackholesimple_dump
subroutine node_component_darkmatterprofile_dump
subroutine node_component_disk_dump
subroutine node_component_disknull_dump
subroutine node_component_formationtime_dump
subroutine node_component_formationtimecole2000_dump
subroutine node_component_formationtimenull_dump
subroutine node_component_hothalonull_dump
subroutine node_component_hothaloverysimple_dump
subroutine node_component_indicesnull_dump
subroutine node_component_interoutputnull_dump
subroutine assign_scale_radii
subroutine create_node_indices
function evolve_to_time
subroutine events_node_merger
tree_node_promote
subroutine satellite_merger_process
subroutine chemicals_dump
subroutine kepler_orbits_dump
subroutine basiccreatelinked
darkmatterprofilecreatelinked
formationtimecreatelinked
indicescreatelinked
mergingstatisticscreatelinked
node_component_basicnonevolving_dump
node_component_basicstandard_dump
node_component_blackholenull_dump
node_component_blackholesimple_dump
node_component_darkmatterprofilenull_dump
node_component_darkmatterprofilescaleshape_dump
node_component_diskexponential_dump
node_component_diskverysimple_dump
node_component_formationtimecole2000_dump
node_component_hothalo_dump
node_component_hothalostandard_dump
node_component_indicesnull_dump
node_component_interoutputnull_dump
subroutine node_component_interoutputstandard_dump
subroutine node_component_mergingstatisticsnull_dump
subroutine node_component_mergingstatisticsstandard_dump
subroutine node_component_mergingstatisticsnull_dump
subroutine node_component_mergingstatisticsrecent_dump
subroutine node_component_position_dump
subroutine node_component_positionnull_dump
subroutine node_component_positionpreset_dump
subroutine node_component_satellite_dump
subroutine node_component_satellitenull_dump
subroutine node_component_satellitepreset_dump
subroutine node_component_satelliteverysimple_dump
subroutine node_component_spheroid_dump
subroutine node_component_spheroidnull_dump
subroutine node_component_spheroidstandard_dump
subroutine node_component_spheroidnull_dump
subroutine node_component_spheroidstandard_dump
subroutine node_component_spinnull_dump
subroutine node_component_spinpreset_dump
subroutine node_component_spinrandom_dump
subroutine node_component_spin_dump
subroutine positioncreatelinked
subroutine spheroidcreatelinked
subroutine tree_node_remove_from_mergee
subroutine node_component_disk_exponential_post_evolve
subroutine node_component_disk_very_simple_post_evolve
subroutine node_component_spheroid_standard_post_evolve
subroutine satellite_merging_remnant_size_cole2000
subroutine satellite_merging_remnant_size_covington2008
subroutine satellite_move_to_new_host
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_rate_tabulate_farahi
function excursion_sets_first_crossing_probability_zhang_hui
function excursion_sets_first_crossing_probability_zhang_hui_high
subroutine make_table
subroutine io_hdf5_close
subroutine alloc_array_character_1d
subroutine alloc_array_double_precision_1d
subroutine alloc_array_double_precision_3d
subroutine alloc_array_double_precision_2d
subroutine alloc_array_double_precision_4d
subroutine alloc_array_double_precision_5d
subroutine alloc_array_integer_2d
subroutine alloc_array_integer_kind_int8_2d
subroutine alloc_array_real_kind_quad_1d
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_kind_int8_1d
subroutine dealloc_array_integer_kind_int8_2d
subroutine dealloc_array_logical_1d
subroutine dealloc_array_real_kind_quad_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
18.1. Program units

**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:** 176
module: galacticus_error

Description: Implements error reporting for the GALACTICUS package.

Code lines: 156

Contained by: file galacticus.error.F90

Modules used:
- hdf5

Used by:
- program tests_excursion_sets
- program halo_mass_functions
- function fgabnd
- subroutine accretion_halos_initialize
- subroutine accretion_disks_adaf_get_parameters
- subroutine atomic_cross_section_ionization_photo_initialize
- function atom_lookup
- subroutine atomic_rate_ionization_collisonal_initialize
- function atomic_rate_recombination_radiative_vernver
- function black_hole_binary_recoil_velocity
- function black_hole_binary_separation_growth_rate_standard
- function black_hole_frame_dragging_frequency_node
- function black_hole_isco_radius_spin
- function black_hole_metric_d_factor_node
- subroutine chemical_hydrogen_rates_compute
- subroutine chemical_state_cie_file_read
- subroutine cooling_function_cie_file_read
- subroutine cooling_radius_initialize
- subroutine cooling_rate_initialize
- subroutine cooling_rate_modifier_cut_off
- subroutine cooling_rate_simple_scaling_initialize
- subroutine freefall_radius_initialize

- program galacticus
- program power_spectra
- subroutine xermsg
- subroutine accretion_halos_simple_initialize
- subroutine accretion_disks_initialize
- function abundance_pattern_lookup
- subroutine atomic_data_initialize
- subroutine atomic_rate_recombination_radiative_initialize
- function black_hole_binary_initial_radius
- function black_hole_binary_separation_growth_rate
- subroutine black_hole_binary_merger
- function black_hole_binary_recoil_velocity
- function black_hole_binary_separation_growth_rate_standard
- function black_hole_isco_radius_spin
- function black_hole_frame_dragging_frequency_node
- function black_hole_isco_radius_spin
- function black_hole_metric_a_factor_node
- function black_hole_metric_d_factor_node
- function black_hole_static_radius_node
- subroutine chemical_hydrogen_rates_initialize
- subroutine chemical_state_initialize
- subroutine cooling_function_initialize
- subroutine cooling_radius_simple_initialize
- subroutine cooling_rate_simple_initialize
- subroutine cooling_rate_white_frenk_initialize
- subroutine cooling_rate_simple_initialize
- subroutine cooling_time_initialize
- subroutine cooling_freefall_time_available_initialize
subroutine freefall_time_available_-halo_formation_initialize
subroutine cooling_specific-angular_-momentum_initialize
subroutine cooling_time_available_-initialize
subroutine cooling_time_available_-halo_formation_initialize
function cmb_temperature_matter_dark_-energy
function comoving_distance_matter_-dark_energy
function cosmology_dark_energy_-equation_of_state_dark_energy
function expansion_factor_matter_dark_-energy
function omega_dark_energy_matter_-dark_energy
function time_from_comoving_distance_-matter_dark_energy
function comoving_distance_conversion_-matter_dark_energy
function cosmology_age_matter_dark_-energy
function cosmology_dark_energy_-exponent_dark_energy
function hubble_parameter_matter_dark_-energy
function omega_matter_total_matter_-dark_energy
function cmb_temperature_matter_lambda
function comoving_distance_conversion_-matter_lambda
function cosmology_age_matter_lambda
function hubble_parameter_matter_lambda
function omega_matter_total_matter_-lambda
function omega_matter
function dark_matter_halo_mass_-accretion_time_zhao2009
subroutine dark_matter_halo_spins_-initialize
subroutine dark_matter_profile_-einstasto_initialize
subroutine dark_matter_profile_nfw_-initialize
subroutine dark_matter_concentrations_-initialize
function dark_matter_profile_-potential_task
subroutine galactic_dynamics_bar_-instability_initialize
subroutine galactic_structure_-enclosed_mass_defaults
subroutine infall_radius_initialize
subroutine cooling_specific_am_mean_-initialize
subroutine cooling_time_available_wf_-initialize
subroutine cosmology_functions_-initialize
function comoving_distance_conversion_-matter_dark_energy
function cosmology_age_matter_dark_-energy
function cosmology_dark_energy_-exponent_dark_energy
function hubble_parameter_matter_dark_-energy
function omega_matter_total_matter_-dark_energy
function cmb_temperature_matter_lambda
function comoving_distance_conversion_-matter_lambda
function cosmology_age_matter_lambda
function hubble_parameter_matter_lambda
function omega_matter_total_matter_-lambda
function omega_matter
subroutine dark_matter_mass_accretion_-initialize
subroutine dark_matter_halo_mass_loss_-rates_initialize
function halo_spin_distribution_sample
subroutine dark_matter_profile_-initialize
function dark_matter_profile_-potential_isothermal
subroutine dark_matter_shapes_-initialize
subroutine events_node_merger_do_slh
function galactic_structure_density
function galactic_structure_radius_-enclosing_mass
function galactic_structure_component_-type_decode
subroutine galactic_structure_radii_-solve
subroutine solve_for_radius

subroutine galactic_structure_radii_-solve_simple
subroutine galacticus_time_per_tree_-initialize
subroutine galacticus_build_output

function filter_lightcone_get_-coordinates
subroutine galacticus_merger_tree_-output_filter_luminosity_initialize
subroutine galacticus_output_star_-formation_histories_initialize
subroutine star_formation_history_-metallicity_split_make_history
function galacticus_output_time_index
subroutine conditional_stellar_mass_-functions_initialize
function hot_halo_profile_radial_-moment_cored_isothermal_get
subroutine hot_halo_density_cored_-isothermal_core_radii_initialize
function hot_halo_ram_pressure_-stripping_radius
subroutine filter_response_load

function intergalactic_medium_-electron_scattering_time
subroutine intergalactic_medium_state_-file_read_data
function hypergeometric_2f1
function generalized_press_schechter_-branch_mass
subroutine tree_branching_modifiers_-initialize
subroutine merger_tree_build_initialize

function node_definition_index
subroutine merger_trees_mass_function_-sampling_initialize

function galactic_structure_mass_type_-decode
subroutine galactic_structure_radii_-solve_adiabatic
subroutine galactic_structure_radii_-initial_initialize
function galactic_structure_surface_-density
subroutine galacticus_time_per_tree_-file_initialize
subroutine galacticus_output_tree_-descendents_initialize
subroutine galacticus_merger_tree_-output_filter_lightcone_initialize
subroutine galacticus_merger_tree_-rotation_curve_initialize
subroutine star_formation_histories_-metallicity_split_initialize
subroutine galacticus_output_tree_-velocity_dispersion_initialize
subroutine galacticus_version_output

subroutine hot_halo_density_clicked
function hot_halo_density_cored_-isothermal_core_radii_initialize
function hot_halo_ram_pressure_force

subroutine hot_halo_temperature_-initialize
function intergalactic_medium_-electron_scattering_optical_depth
subroutine intergalactic_medium_state_-initialize
function inverse_gamma_function_-incomplete_complementary
subroutine tree_branching_initialize
function generalized_press_schechter_-subresolution_fraction
function merger_tree_create

subroutine merger_tree_construct_-fully_specified
function node_lookup
subroutine assign_mergers
18.1. Program units

subroutine assign_scale_radii
subroutine build_isolated_parent_pointers
subroutine build_subhalo_mass_histories
subroutine merger_tree_read_do
subroutine read_node_data
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 abundances_atomic_index
function abundances_get_metallicity
subroutine abundances_set_metallicity
function chemicals_names
subroutine chemical_structure_initialize
function coordinates_null_to
subroutine history_builder
subroutine history_create
subroutine history_increment
subroutine history_trim
subroutine kepler_orbits_builder
subroutine kepler_orbits_propagate
function kepler_orbits_specific_reduced_mass
function kepler_orbits_velocity_scale
function mass_distribution_create
function mass_distribution_density_radial_moment_null
function mass_distribution_mass_enc_by_sphere_null
subroutine merger_tree_data_construct_particle_indices
subroutine merger_tree_data_structure_export
subroutine build_descendent_pointers
subroutine build_parent_pointers
subroutine enforce_subhalo_status
subroutine merger_tree_read_initialize
subroutine scan_for_mergers
function evolve_to_time
subroutine events_node_merge
subroutine merger_trees_millennium_process
subroutine merger_tree_regrid_time
subroutine odeiv2_solve
function interpolate
function interpolate_locate
function make_range
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
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_density_null
function mass_distribution_half_mass_radius_spherical
function mass_distribution_potential_null
subroutine merger_tree_data_structure_add_metadata
subroutine merger_tree_data_structure_export_rate
subroutine merger_tree_data_structure_read_ascii
subroutine merger_tree_data_structure_set_property_double
subroutine merger_tree_data_structure_set_units
module galacticus_nodes
function blackholeget
subroutine diskabundancesgasrate
subroutine diskexponentialabundancesgasrategeneric
subroutine diskexponentialmassgasrategeneric
subroutine diskmassgasrate
function formationtimeget
function indicesget
function interoutputget
function mergingstatisticsget
subroutine node_component_basic_initializor
subroutine node_component_basic_remove
subroutine node_component_basicnull_builder
subroutine node_component_blackhole_initializor
subroutine node_component_blackhole_remove
subroutine node_component_blackholenull_builder
subroutine node_component_blackholesimple_builder
subroutine node_component_blackholestandard_builder
darkmatterprofile_initializor
subroutine node_component_darkmatterprofile_remove
subroutine node_component_darkmatterprofilescaleshape_builder
subroutine node_component_darkmatterprofile_builder
subroutine node_component_disk_initializor
subroutine node_component_disk_remove
subroutine node_component_disknull_builder
subroutine node_component_diskbuilder
subroutine node_component_-diskverysimple_builder
subroutine node_component_-formationtime_initializor
subroutine node_component_-formationtime_remove
subroutine node_component_-formationtime_initializor
subroutine node_component_-formationtime_move
subroutine node_component_-formationtimecole2000_builder
subroutine node_component_hothalo_-builder
subroutine node_component_hothalo_-initializor
subroutine node_component_hothalo_-remove
subroutine node_component_hothalo_-initializor
subroutine node_component_hothalo_-remove
subroutine node_component_-hothalostandard_builder
subroutine node_component_-hothaloverysimple_builder
subroutine node_component_indices_-builder
subroutine node_component_indices_-initializor
subroutine node_component_indices_-move
subroutine node_component_indices_-remove
subroutine node_component_indicesnull_-builder
subroutine node_component_-indicesstandard_builder
subroutine node_component_-interoutput_-builder
subroutine node_component_-interoutput_-initializor
subroutine node_component_-interoutput_-move
subroutine node_component_-interoutput_-remove
subroutine node_component_-interoutputnull_builder
subroutine node_component_-interoutputstandard_builder
subroutine node_component_-mergingstatistics_builder
subroutine node_component_-mergingstatistics_initializor
subroutine node_component_-mergingstatistics_move
subroutine node_component_-mergingstatisticsnull_builder
subroutine node_component_-mergingstatisticsrecent_builder
subroutine node_component_-mergingstatisticsstandard_builder
subroutine node_component_position_-initializer
subroutine node_component_position_-remove
subroutine node_component_position_-positionnull_builder
subroutine node_component_satellite_-builder
subroutine node_component_satellite_-initializer
subroutine node_component_satellite_-move
subroutine node_component_satellite_-satellitenull_builder
subroutine node_component_satellitepreset_builder
subroutine node_component_satellitestandard_builder
subroutine node_component_satelliteverysimple_builder
subroutine node_component_spheroid_initializer
subroutine node_component_spheroid_remove
subroutine node_component_spheroidnull_builder
subroutine node_component_spheroidstandard_builder
subroutine node_component_spin_initializer
subroutine node_component_spin_remove
subroutine node_component_spinningpreset_builder
subroutine node_component_spinnull_builder
function positionget
subroutine spheroidabundancesgasrate
subroutine spheroidabundancesstellarrate
subroutine spheroidangularmomentumrate
function spheroidget
subroutine spheroidluminositiesstellarrate
subroutine spheroidmassgasrate
subroutine spheroidmassstellarrate
subroutine spheroidstandardabundancesgasrategeneric
subroutine spheroidstandardabundancesstellarrategeneric
subroutine spheroidstandardangularmomentumrategeneric
subroutine spheroidstandardluminositiesstellarrategeneric
subroutine spheroidstandardmassgasrategeneric
subroutine spheroidstandardmassstellarrategeneric
subroutine spheroidstandardstellarpropertieshistoryrategeneric
function spinget
subroutine treenodeinitialize
subroutine node_component_basic_non_evolving_promote
subroutine node_component_dark_matter_profile_scale_promote
subroutine node_component_disk_profile_scale_shape_promote
subroutine node_component_disk_exponential_satellite_merging
subroutine node_component_hot_halo_standard_heat_source
subroutine node_component_hot_halo_standard_initialize
subroutine node_component_inter_output_standard_satellite_merger
subroutine node_component_merging_statistics_recent_node_merger
subroutine node_component_spheroid_standard_initialize
subroutine node_component_spin_random_promote
subroutine table_1d_reverse
subroutine table_generic_1d_populate_single
subroutine table_linear_1d_populate_single
subroutine table_linear_cspline_1d_populate_single
subroutine table_linear_cspline_1d_populate
subroutine radiation_initialize_intergalactic_background
function ram_pressure_stripping_mass_loss_rate_disk
subroutine satellite_merging_mass_movement
subroutine satellite_merging_remnant_size_cole2000
subroutine satellite_merging_remnant_size_covington2008
subroutine satellite_merging_remnant_progenitor_properties_cole2000
subroutine satellite_merging_remnant_progenitor_properties
subroutine satellite_merging_remnant_progenitor_properties_standard_init
function virial_orbital_parameters_benson2005
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 node_component_merging_statistics_recent_initialize
subroutine node_component_spheroid_standard_energy_gas_input_rate
subroutine node_component_spheroid_standard_mass_gas_sink_rate
subroutine node_component_spin_preset_promote
function table1d_find_effective_x
subroutine table_generic_1d_populate
subroutine table_linear_1d Populate
subroutine radiation_initialize_intergalactic_background
function ram_pressure_stripping_mass_loss_rate_disk
subroutine satellite_merging_mass_movement
subroutine satellite_merging_remnant_size_cole2000
subroutine satellite_merging_remnant_size_covington2008
subroutine satellite_merging_remnant_progenitor_properties_cole2000
subroutine satellite_merging_remnant_progenitor_properties_standard
subroutine satellite_merging_timescales_initialize
function virial_orbital_parameters
function imf_energy_input_rate_noninstantaneous
function imf_name
subroutine star_formation_imf_initialize
function imf_select_disk_spheroid
subroutine star_formation_feedback_disks_initialize
subroutine star_formation_feedback_expulsive_feedback_disks_initialize
subroutine star_formation_expulsive_feedback_spheroids_initialize
subroutine star_formation_rate_surface_density_disks_initialize
subroutine star_formation_timescale_disks_dynamical_time_initialize
subroutine stellar_astrophysics_initialize
subroutine stellar_astrophysics_file_initialize
subroutine supernovae_population_iii_hegerwoosley_initialize
subroutine supernovae_population_iii_nagashima_initialize
subroutine stellar_tracks_initialize_file
function stellar_population_luminosity
function stellar_population_luminosities_index
subroutine stellar_population_properties_luminosities_initialize
function stellar_population_spectra_file_interpolate
function stellar_population_spectrum_postprocess_index
function critical_overdensity_forCollapse
critical_overdensity_forCollapseTimeGradient
subroutine critical_overdensity_massScalingInitialize
subroutine critical_overdensity_massScalingWDMInitialize
subroutine excursion_sets_first_crossings_initialize
function excursion_sets_non_crossing_rate_zhang_hui
function excursion_sets_non_crossing_rate_zhang_hui_high
subroutine halo_mass_function_initialize
function linear_growth_factor
subroutine linear_growth_initialize
subroutine star_formation_rate_surface_density_disks_br_initialize
subroutine star_formation_timescale_disks_initialize
subroutine star_formation_timescale_spheroids_initialize
subroutine stellar_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_luminosities_output
subroutine stellar_population_spectrum_file_initialize
function critical_overdensity_forCollapse
subroutine critical_overdensity_massScalingInitialize
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

subroutine power_spectrum_nonlinear_-cosmicemu

function power_spectrum_nonlinear_-peacockdodds1996

subroutine power_spectrum_window_-functions_initialize

subroutine make_table

subroutine transfer_function_file_read

function halo_virial_density_contrast

subroutine virial_density_bryan_-norman_initialize

function halo_virial_density_contrast_-rate_of_change

subroutine virial_density_kitayama_-suto1996_initialize

subroutine system_command_do

subroutine io_hdf5_assert_attribute_type

function io_hdf5_character_types

subroutine io_hdf5_create_reference_-scalar_to_1d

subroutine io_hdf5_create_reference_-scalar_to_3d

subroutine io_hdf5_create_reference_-scalar_to_5d

subroutine io_hdf5_read_attribute_-character_1d_array_allocatable

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_-integer8_1d_array_static

subroutine io_hdf5_read_attribute_-integer_1d_array_allocatable

subroutine io_hdf5_read_attribute_-integer_1d_array_static

subroutine power_spectrum_initialize

subroutine power_spectrum_nonlinear_-initialize

subroutine power_spectrum_nonlinear_-peacockdodds1996

subroutine power_spectrum_window_-functions_initialize

subroutine make_table

subroutine transfer_function_file_read

function halo_virial_density_contrast

subroutine virial_density_bryan_-norman_initialize

function halo_virial_density_contrast_-rate_of_change

subroutine virial_density_kitayama_-suto1996_initialize

subroutine system_command_do

subroutine io_hdf5_assert_attribute_type

function io_hdf5_character_types

subroutine io_hdf5_create_reference_-scalar_to_1d

subroutine io_hdf5_create_reference_-scalar_to_3d

subroutine io_hdf5_create_reference_-scalar_to_5d

function io_hdf5_dataset_size

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_allocatable

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_-integer8_1d_array_static

subroutine io_hdf5_read_attribute_-integer_1d_array_allocatable

subroutine io_hdf5_read_attribute_-integer_1d_array_static

subroutine io_hdf5_read_attribute_-varstring_1d_array_allocatable

subroutine io_hdf5_read_attribute_-varstring_1d_array_static
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_varstring_1d_array_allocatable
subroutine io_hdf5_write_attribute_character_1d
subroutine io_hdf5_write_attribute_double_1d
subroutine io_hdf5_write_attribute_integer8_1d
subroutine io_hdf5_write_attribute_character_1d
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_hdf_assert_is_initialized

function xml_get_first_element_by_tag_name
function array_index_double_2d
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
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_kind_quad_1d
subroutine assert_logical_scalar

**subroutine:** galacticus_error_handler_register
*Description:* Register signal handlers.
*Code lines:* 13
*Contained by:* module galacticus_error
*Modules used:* fgsl

**interface:** galacticus_error_report
*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:* 19
*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:* 24
*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:* 17
*Contained by:* module galacticus_error

**subroutine:** galacticus_signal_handler_sigint
*Description:* Handle SIGINT signals, by flushing all data and then aborting.
*Code lines:* 17
*Contained by:* module galacticus_error
subroutine: galacticus_signal_handler_sigsegv
Description: Handle SIGSEGV signals, by flushing all data and then aborting.
Code lines: 17
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: subroutine atomic_data_initialize, subroutine chemical_state_atomic_cie_cie_cloudy_create, subroutine galacticus_build_output, subroutine cooling_function_atomic_cie_cie_cloudy_create, subroutine filter_response_load, subroutine intergalactic_medium_state_recfast_initialize, subroutine chemical_structure_initialize, subroutine imf_energy_input_rate_noninstantaneous, function imf_metal_yield_rate_noninstantaneous, subroutine imf_recycling_rate_noninstantaneous, subroutine supernovae_type_ia_nagashima_initialize, 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_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
18.1. Program units

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
Used by: program optimal_sampling_smf function merger_tree_construct_mass-
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
18. Source Code Documentation

**iso_varying_string**

**file: galacticus.meta.evolver_profiler.F90**
*Description:* Contains a module which constructs a profile of GALACTICUS ODE evolver statistics.
*Code lines:* 157

**module: galacticus_meta_evolver_profiler**
*Description:* Constructs a profile of GALACTICUS ODE evolver statistics.
*Code lines:* 137
*Contained by:* file galacticus.meta.evolver_profiler.F90
*Modules used:* hashes
*Used by:* subroutine galacticus_output_close_file

**subroutine: galacticus_meta_evolver_profile**
*Description:* Record profiling information on the ODE evolver.
*Code lines:* 73
*Contained by:* module galacticus_meta_evolver_profiler
*Modules used:* arrays_search input_parameters
iso_varying_string memory_management
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
18.1. Program units

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
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: 39

module: galacticus_hdf5
Description: Manages HDF5 output from GALACTICUS.
Code lines: 19
Contained by: file galacticus.output.HDF5.F90
Modules used: hdf5

Used by:

- subroutine meta_tree_timing_output
- subroutine galacticus_build_output
- subroutine galacticus_extra_output_-halo_fourier_profile
- subroutine star_formation_history_-_output_metallicity_split
- subroutine merger_tree_history_write
- subroutine merger_tree_mass_accretion_-_history_output
- subroutine node_component_black_hole_-standard_output_merger

- subroutine galacticus_meta_evolver_-profiler_output
- module galacticus_output_open
- module galacticus_output_merge_tree
- subroutine galacticus_linear_power_-_spectrum_output
- subroutine galacticus_version_output
- subroutine merger_tree_record_-_evolution_output
- subroutine merger_tree_structure_output
- subroutine node_component_black_hole_-standard_output_properties

file: galacticus.output.HDF5.open.F90
Description: Contains a module which handles opening of the GALACTICUS output file.
Code lines: 168

module: galacticus_output_open
18. Source Code Documentation

**Description:** Handles opening of the GALACTICUS output file.

**Code lines:** 148

**Contained by:** file `galacticus.output.HDF5.open.F90`

**Modules used:** `galacticus_hdf5`  
`iso_varying_string`

**Used by:** subroutine `galacticus_merger_tree_output`  
function `galacticus_task_end`

**subroutine:** `galacticus_output_close_file`

**Description:** Close the GALACTICUS output file.

**Code lines:** 43

**Contained by:** module `galacticus_output_open`

**Modules used:** `galacticus_meta_evolver_profiler`  
`galacticus_meta_tree_timing`  
`input_parameters`  
`merger_tree_mass_accretion_history`  
`merger_tree_output_structure`  
`merger_tree_timesteps_history`

**subroutine:** `galacticus_output_open_file`

**Description:** Open the file for GALACTICUS output.

**Code lines:** 88

**Contained by:** module `galacticus_output_open`

**Modules used:** `galacticus_build`  
`galacticus_output_halo_models`  
`galacticus_versioning`  
`input_parameters`

**file:** `galacticus.output.build.F90`

**Description:** Contains a module which implements writing of GALACTICUS build information to the GALACTICUS output file.

**Code lines:** 126

**module:** `galacticus_build`

**Description:** Implements writing of GALACTICUS build information to the GALACTICUS output file.

**Code lines:** 103

**Contained by:** file `galacticus.output.build.F90`

**Modules used:** `iso_c_binding`  
`fox_common`  
`galacticus_error`  
`galacticus_hdf5`  
`galacticus_input_paths`  
`iso_varying_string`  
`string_handling`

**subroutine:** `galacticus_build_output`

**Description:** Output build information to the main output file.

**Code lines:** 81

**Contained by:** module `galacticus_build`

**Modules used:** `fgsl`  
`file_utilities`  
`fox_common`  
`galacticus_error`  
`galacticus_hdf5`  
`galacticus_input_paths`  
`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:** 700

**module:** `galacticus_output_merger_tree`
18.1. Program units

**Description:** Implements writing a merger tree to the GALACTICUS output file.

**Code lines:** 680

**Contained by:** file `galacticus.output.merger_tree.F90`

**Modules used:** `galacticus_hdff`, `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:** 30

**Contained by:** module `galacticus.output.merger_tree`

**Modules used:** `memory_management`

**subroutine:** count_properties

**Description:** Count up the number of properties that will be output.

**Code lines:** 68

**Contained by:** module `galacticus.output.merger_tree`

**Modules used:** `cooling_radii`, `cooling_rates`, `galacticus_merger_tree_output_filter_lightcones`, `galacticus_output_halos_models`, `galacticus_output_most_massive_progenitors`, `galacticus_output_tree_half_light_properties`, `galacticus_output_trees_density_contrasts`, `galacticus_output_trees_links`, `galacticus_output_trees_main_branch`, `galacticus_output_trees_rotation_curve`, `galacticus_output_trees_velocity_dispersion`, `galacticus_output_trees_virial`, `node_component_black_hole_simple`, `node_component_black_hole_standard`, `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:** 24

**Contained by:** module `galacticus.output.merger_tree`

**subroutine:** establish_property_names

**Description:** Set names for the properties.

**Code lines:** 70

**Contained by:** module `galacticus.output.merger_tree`

**Modules used:** `cooling_radii`, `cooling_rates`, `galacticus_merger_tree_output_filter_lightcones`, `galacticus_output_halos_models`, `galacticus_output_most_massive_progenitors`, `galacticus_output_tree_half_light_properties`, `galacticus_output_trees_density_contrasts`, `galacticus_output_trees_links`, `galacticus_output_trees_main_branch`, `galacticus_output_trees_rotation_curve`, `galacticus_output_trees_velocity_dispersion`, `galacticus_output_trees_virial`, `node_component_black_hole_simple`, `node_component_black_hole_standard`, `node_component_merging_statistics_recent`
subroutine: galacticus_merger_tree_output
Description: Write properties of nodes in thisTree to the GALACTICUS output file.
Code lines: 261
Contained by: module galacticus_output_merger_tree
Modules used:

- cooling_radii
- galactic_structure_radii
- galacticus_merger_tree_output_filters
- galacticus_output_halos
- galacticus_output_open
- galacticus_output_tree_mass_profiles
- galacticus_output_trees_descendents
- galacticus_output_trees_links
- galacticus_output_trees_main_branch
- galacticus_output_trees_rotation_curve
- galacticus_output_trees_satellite_pericenter
- galacticus_output_trees_velocity_dispersion
- galacticus_output_trees_virial
- input_parameters
- merger_tree_timesteps_record_evolution
- node_component_black_hole_simple
- node_component_black_hole_standard
- node_component_inter_output_standard
- node_component_merging_statistics_recent
- node_component_spheroid_standard

subroutine: galacticus_merger_tree_output_finalize
Description: Finalize merger tree output by closing any open groups.
Code lines: 16
Contained by: module galacticus_output_merger_tree

subroutine: galacticus_merger_tree_output_make_group
18.1. Program units

*Description:* Make an group in the GALACTICUS file in which to store thisTree.

*Code lines:* 25

*Contained by:* module galacticus_output_merger_tree

*Modules used:* galacticus_nodes 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:* 24

*Contained by:* module galacticus_output_merger_tree

**subroutine: make_output_group**

*Description:* Create a group in which to store this output.

*Code lines:* 99

*Contained by:* module galacticus_output_merger_tree

*Modules used:* cosmology_functions galacticus_output_halo_models

memory_management numerical_constants_astronomical

string_handling

**type: outputgroup**

*Description:* Type used for output group information.

*Code lines:* 5

*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:* 257

**module: galacticus_output_trees_density_contrasts**

*Description:* Handles outputting of node density contrast properties (radii and masses).

*Code lines:* 237

*Contained by:* file galacticus_output_merger_tree.density_contrasts.F90

*Modules used:* galacticus_nodes

*Used by:* subroutine count_properties subroutine establish_property_names

subroutine galacticus_merge_tree_output

**subroutine: galacticus_output_tree_density_contrast**

*Description:* Store density contrast properties in the GALACTICUS output file buffers.

*Code lines:* 47

*Contained by:* module galacticus_output_trees_density_contrasts

*Modules used:* dark_matter_halo_scales galactic_structure_enclosed_masses

galactic_structure_options kind_numbers

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:* 80
Contained by: module galacticus_output_trees_density_contrasts
Modules used: cosmological_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: 14
Contained by: module galacticus_output_trees_density_contrasts
Modules used: cosmology_functions galactic_structure_options galactic_structure_enclosed_masses
               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: 218

module: galacticus_output_trees_descendents
Description: Handles outputting of tree descendent data to the GALACTICUS output file.
Code lines: 198
Contained by: file galacticus.output.merger_tree.descendents.F90
Modules used: galacticus_nodes
terminated

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_descendents_initialize
Description: Initializes the module by determining whether or not descendent data should be output.
Code lines: 31
Contained by: module galacticus_output_trees_descendents
Modules used: galacticus_error input_parameters
**subroutine: galacticus_output_tree_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_descendents

**subroutine: galacticus_output_tree_descendents_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:** 144

**module: galacticus_merger_tree_output_filters**
- **Description:** Provides filtering of output.
- **Code lines:** 124
- **Contained by:** file galacticus.output.merger_tree.filters.F90
- **Modules used:** iso_varying_string
- **Used by:** subroutine galacticus_merger_tree_output_output

**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:** 41
- **Contained by:** module galacticus_merger_tree_output_filters
- **Modules used:** galacticus_merger_tree_output_filter_lightcones galacticus_merger_tree_output_filter_luminosities galacticus_merger_tree_output_filter_stellar_masses galacticus_nodes

**subroutine: galacticus_merger_tree_output_filter_initialize**
- **Description:** Initialize the output filter subsystem.
- **Code lines:** 61
- **Contained by:** module galacticus_merger_tree_output_filters
- **Modules used:** galacticus_merger_tree_output_filter_lightcones galacticus_merger_tree_output_filter_luminosities galacticus_merger_tree_output_filter_stellar_masses input_parameters memory_management

**file: galacticus.output.merger_tree.filters.lightcone.F90**
- **Description:** Contains a module which filters output for lightcone geometry.
- **Code lines:** 449

**module: galacticus_merger_tree_output_filter_lightcones**
Description: Filters output for lightcone geometry.

Code lines: 429

Contained by: file `galacticus.output.merger_tree.filters.lightcone.F90`

Used by: subroutine `count_properties` subroutine `establish_property_names` subroutine `galacticus_merger_tree_output` subroutine `galacticus_merger_tree_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` `io_xml`

subroutine: `galacticus_merger_tree_output_filter_lightcone`

Description: Determines whether `thisNode` lies within a lightcone and, therefore, should be output.

Code lines: 73

Contained by: module `galacticus_merger_tree_output_filter_lightcones`

Modules used: `arrays_search` `cosmology_functions` `galacticus_nodes`

subroutine: `galacticus_merger_tree_output_filter_lightcone_initialize`

Description: Initializes the lightcone filter module.

Code lines: 127

Contained by: module `galacticus_merger_tree_output_filter_lightcones`

Modules used: `cosmological_parameters` `cosmology_functions` `fox_dom` `galacticus_error` `input_parameters` `io_xml` `iso_varying_string` `memory_management` `numerical_constants_astronomical` `vectors`

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
18.1. Program units

**Contained by:** module `galacticus_merger_tree_output_filter_lightcones`  
**Modules used:** `galacticus_nodes`

**file:** `galacticus.output.merger_tree.filters.luminosities.F90`  
**Description:** Contains a module which filters output on stellar mass.  
**Code lines:** 137

**module:** `galacticus_merger_tree_output_filter_luminosities`  
**Description:** Filters output for lightcone geometry.  
**Code lines:** 117  
**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_population_properties_luminosities`

**subroutine:** `galacticus_merger_tree_output_filter_luminosity_initialize`  
**Description:** Initializes the stellar mass filter module.  
**Code lines:** 40  
**Contained by:** module `galacticus_merger_tree_output_filter_luminosities`  
**Modules used:** `galacticus_error`  
`input_parameters`  
`iso_varying_string`  
`memory_management`  
`stellar_population_properties_luminosities`

**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.half_light_properties.F90
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_population_properties_luminosities

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 stellar_population_properties_luminosities

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_population_properties_luminosities

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
18.1. Program units

**Modules used:** stellar_population_properties_-luminosities

**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:* 396

**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:* 376

*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:* 68

*Contained by:* module galacticus_output_halo_models

*Modules used:* cosmology_functions  dark_matter_profiles  galacticus_hdf5  galacticus_nodes  iso_varying_string  kind_numbers  memory_management  string_handling

**subroutine:** galacticus_growth_factor_output

*Description:* Output the linear theory power spectrum to the main output file.

*Code lines:* 19

*Contained by:* module galacticus_output_halo_models

*Modules used:* io_hdf5  linear_growth

**subroutine:** galacticus_linear_power_spectrum_output

*Description:* Output the linear theory power spectrum to the main output file.

*Code lines:* 78

*Contained by:* module galacticus_output_halo_models

*Modules used:* galacticus_hdf5  memory_management  numerical_constants_astronomical  numerical_ranges  power_spectra

**subroutine:** galacticus_output_halo_model

*Description:* Store halo model properties in the GALACTICUS output file buffers.

*Code lines:* 28

*Contained by:* module galacticus_output_halo_models

*Modules used:* dark_matter_halo_biases  galacticus_nodes  kind_numbers
subroutine: galacticus_output_halo_model_initialize
Description: Initializes the module by determining whether or not halo model data should be output.
Code lines: 64
Contained by: module galacticus_output_halo_models
Modules used: input_parameters

subroutine: galacticus_output_halo_model_names
Description: Set the names of halo model properties to be written to the GALACTICUS output file.
Code lines: 41
Contained by: module galacticus_output_halo_models
Modules used: galacticus_nodes

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: 17
Contained by: module galacticus_output_halo_models
Modules used: galacticus_nodes

file: galacticus.output.merger_tree.links.F90
Description: Contains a module which handles outputting of tree link data to the GALACTICUS output file.
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
18.1. Program units

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
Contained 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
Contained 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
Contained 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
Contained by: file galacticus.output.merger_tree.mass_profile.F90
Modules used: galacticus_nodes
Used by: subroutine count_properties subroutine establish_property_names
18. Source Code Documentation

**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
- **Contained 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
- **Contained 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
- **Contained by:** module galacticus_output_tree_mass_profiles
- **Modules used:** numerical_constants_astronomical

**subroutine** galacticus_output_tree_mass_profile_property_count
- **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.
18.1. Program units

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.rotation_curve.F90
Description: Contains a module which handles outputting of rotation curve data to the GALACTICUS output file.
Code lines: 310

module: galacticus_output_trees_rotation_curve
Description: Handles outputting of rotation curve data to the GALACTICUS output file.
Code lines: 290
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: 56
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 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: galactic_structure_options galacticus_error galacticus_nodes input_parameters stellar_population_properties_luminosities

subroutine: galacticus_output_tree_rotation_curve_names
Description: Set the names of rotation curve properties to be written to the GALACTICUS output file.
18. Source Code Documentation

**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_pericenter.F90`  
**Description:** Contains a module which handles outputting of satellite orbital pericenter data to the GALACTICUS output file.  
**Code lines:** 183

**module:** `galacticus_output_trees_satellite_pericenter`  
**Description:** Handles outputting of satellite orbital pericenter data to the GALACTICUS output file.  
**Code lines:** 163  
**Contained by:** file `galacticus.output.merger_tree.satellite_pericenter.F90`  
**Used by:** subroutine `count_properties` subroutine `establish_property_names` subroutine `galacticus_merger_tree_output`  
**Modules used:** `galacticus_output_trees_satellite_pericenter` `kind_numbers` `satellite_orbits`  
**subroutine:** `galacticus_output_tree_satellite_pericenter`  
**Description:** Store satellite orbital pericenter properties in the GALACTICUS output file buffers.  
**Code lines:** 43  
**Contained by:** module `galacticus_output_trees_satellite_pericenter`  
**Modules used:** `galacticus_output_trees_satellite_pericenter` `kepler_orbits` `kind_numbers` `satellite_orbits`

**subroutine:** `galacticus_output_tree_satellite_pericenter_initialize`  
**Description:** Initializes the module by determining whether or not satellite pericenter data should be output.  
**Code lines:** 27  
**Contained by:** module `galacticus_output_trees_satellite_pericenter`  
**Modules used:** `input_parameters`

**subroutine:** `galacticus_output_tree_satellite_pericenter_names`  
**Description:** Set the names of satellite orbital pericenter properties to be written to the GALACTICUS output file.  
**Code lines:** 42  
**Contained by:** module `galacticus_output_trees_satellite_pericenter`  
**Modules used:** `galacticus_nodes` `numerical_constants_astronomical`

**subroutine:** `galacticus_output_tree_satellite_pericenter_property_count`
18.1. Program units

Description: Account for the number of satellite orbital pericenter properties to be written to the GALACTICUS output file.

Code lines: 14

Contained by: module `galacticus_output_trees_satellite_pericenter`

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: 209

module: `galacticus_output_star_formation_histories`

Description: Handles computation and output of star formation histories for galaxies.

Code lines: 189

Contained by: file `galacticus.output.merger_tree.star_formation.F90`

Modules used: `abundances_structure`, `galacticus_nodes`, `histories`, `iso_varying_string`, `kind_numbers`

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`, `star_formation_histories_-metallicity_split`

Subroutine: `star_formation_history_create`

Description: Create any history required for storing the star formation history.

Code lines: 13

Contained by: module `galacticus_output_star_formation_histories`

Subroutine: `star_formation_history_output`

Description: Output the star formation history for `thisNode`.

Code lines: 17

Contained by: module `galacticus_output_star_formation_histories`

Subroutine: `star_formation_history_record`

Description: Record the star formation history for `thisNode`.
18. Source Code Documentation

Code lines: 15
Contained by: module galacticus_output_star_formation_histories

subroutine: star_formation_history_scales
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: 475

module: star_formation_histories_metallicity_split
Description: Handles computation and output of star formation histories split by metallicity for galaxies.
Code lines: 455
Contained by: file galacticus.output.merger_tree.star_formation.metallicity_split.F90
Used by: subroutine galacticus_output_star_formation_histories_initialize

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

subroutine: star_formation_history_metallicity_split_make_history
Description: Output the star formation history for thisNode.
Code lines: 84
Contained by: module star_formation_histories_metallicity_split
Modules used: galacticus_hdf5 galacticus_nodes galacticus_output_times histories
subroutine: star_formation_history_record_metallicity_split
Description: Record the star formation history for thisNode.
Code lines: 37
Contained by: module star_formation_histories_metallicity_split
Modules used: abundances_structure arrays_search galacticus_nodes histories

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: 109

module: star_formation_histories_null
Description: Implements a null method for star formation histories.
Code lines: 89
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: 15
Contained by: module star_formation_histories_null
Modules used: galacticus_nodes histories
18. Source Code Documentation

**kind_numbers**

**subroutine: star_formation_history_record_null**
*Description:* Record the star formation history for thisNode.
*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.velocity_dispersion.F90**
*Description:* Contains a module which handles outputting of velocity dispersion data to the GALACTICUS output file.
*Code lines:* 510

**module: galacticus_output_trees_velocity_dispersion**
*Description:* Handles outputting of velocity dispersion data to the GALACTICUS output file.
*Code lines:* 490
*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_velocityDispersion**
*Description:* Store velocity dispersion properties in the GALACTICUS output file buffers.
*Code lines:* 114
*Contained by:* module galacticus_output_trees_velocity_dispersion
*Modules used:* dark_matter_halo_scales, fgs1, galactic_structure_enclosed_masses, galactic_structure_options, iso_c_binding, dispersions, kind_numbers, numerical_integration

**subroutine: galacticus_output_tree_velocityDispersion_init**
*Description:* Initializes the module by determining whether or not velocity dispersion should be output.
*Code lines:* 153
*Contained by:* module galacticus_output_trees_velocity_dispersion
*Modules used:* galactic_structure_options, galacticus_error, input_parameters, stellar_population_properties_luminosities, string_handling
18.1. Program units

**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_dispersion_vlcty_dnsty_intgrnd**
*Description:* Integrand function used for computing line-of-sight velocity dispersions.
*Code lines:* 16
*Contained by:* module `galacticus_output_trees_velocity_dispersion`
*Modules used:* `galactic_structure_densities` `galactic_structure_velocity_dispersions` `iso_c_binding`

**function: galacticus_output_trees_vlcty_dsprsn_dnsty_srfc_intgrnd**
*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_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`
type: radiusspecifier  
Code lines: 5  
Contained by: module galacticus_output_trees_velocity_dispersion

function: spherical_shell_solid_angle_in_cylinder  
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  
Description: Contains a module which handles outputting of node virial data to the GALACTICUS output file.  
Code lines: 163

module: galacticus_output_trees_virial  
Description: Handles outputting of node virial data to the GALACTICUS output file.  
Code lines: 143  
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: 23  
Contained by: module galacticus_output_trees_virial  
Modules used: dark_matter_halo_scales galacticus_nodes kind_numbers

subroutine: galacticus_output_tree_virialInitialize  
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_properties  
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
18.1. Program units

file: galacticus.output.times.F90
Description: Contains a module which provides output times.
Code lines: 176

module: galacticus_output_times
Description: Provides output times.
Code lines: 156
Contained by: file galacticus.output.times.F90
Used by:
  subroutine galacticus_output_tree_descendants
  subroutine star_formation_history_descendants
  subroutine star_formation_history_metallicity_split_make_history
  function galacticus_task_evolve_tree
  subroutine merger_tree_record_evolution_output
  subroutine node_component_merging_statistics_recent_initializer
  subroutine node_component_merging_statistics_recent_node_merger
  subroutine node_component_merging_statistics_recent_output

function: galacticus_next_output_time
Description: Returns the time of the next output after currentTime.
Code lines: 18
Contained by: module galacticus_output_times
Modules used: arrays_search
galacticus_error
numerical_comparison

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: 10
Contained by: module galacticus_output_times

function: galacticus_output_time_index
Description: Returns the index of the output given the corresponding time.
Code lines: 14
Contained by: module galacticus_output_times
Modules used: arrays_search
galacticus_error
numerical_comparison

function: galacticus_previous_output_time
Description: Returns the time of the previous output prior to currentTime.
18. Source Code Documentation

**Code lines:** 19  
**Contained by:** module `galacticus_output_times`  
**Modules used:** `arrays_search`

**subroutine:** `output_times_initialize`  
**Description:** Initialize the output times.  
**Code lines:** 53  
**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:** 108

**module:** `galacticus_versioning`  
**Description:** Implements writing of the version number and run time to the GALACTICUS output file.  
**Code lines:** 88  
**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:** 54  
**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:** 376

**module:** `galacticus_state`  
**Description:** Implements storage and recovery of the Galacticus internal state. Used for restoring random number generator sequences for example.  
**Code lines:** 355  
**Contained by:** file `galacticus.state.F90`  
**Modules used:** `iso_varying_string`
18.1. Program units

**Used by:**
- subroutine `merger_tree_build_do`
- subroutine `merger_tree_state_restore`
- subroutine `merger_tree_read_do`
- subroutine `merger_tree_state_store`

**subroutine: galacticus_state_retrieve**

**Description:** Retrieve the internal state.

**Code lines:** 125

**Contained by:** module `galacticus_state`

**Modules used:**
- `black_hole_binary_recoil_velocities_standard`
- `cosmology_functions_matter_dark_energy`
- `critical_overdensity`
- `dark_matter_profiles_einasto`
- `dynamical_friction_laceycole_tormen`
- `halo_spin_distributions_bett2007`
- `histories`
- `intergalactic_medium_state`
- `linear_growth`
- `linear_growth_simple`
- `merger_tree_build_cole2000`
- `node_component_disk_exponential_data`
- `primordial_power_spectrum_power_law`
- `sphericalcollapse_matter_lambda`
- `transfer_function_bbks`
- `transfer_functions`
- `virial_orbits_benson2005`

**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`
- `dynamical_friction_laceycole_tormen`
- `halo_spin_distributions_bett2007`
- `merger_tree_build_cole2000`
- `virial_orbits_benson2005`

**subroutine: galacticus_state_store**

**Description:** Store the internal state.

**Code lines:** 120

**Contained by:** module `galacticus_state`

**Modules used:**
- `black_hole_binary_recoil_velocities_standard`
- `cosmological_mass_variance_filtered_power_spectrum`
- `cosmology_functions_matter_lambda`
- `dark_matter_halo_scales`
- `dark_matter_profiles_nfw`
- `fgsl`
- `halo_spin_distributions_lognormal`
- `halo_spin_distributions_bett2007`

779
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_merger_tree galacticus_output_open

function: galacticus_task_start
18.1. Program units

**Code lines:** 34  
**Contained by:** module `galacticus_tasks_basic`  
**Modules used:** `galacticus_output_open`  

**file:** `galacticus.tasks.evolve_tree.F90`  
**Description:** Contains a module which implements the task of evolving merger trees.  
**Code lines:** 333

**module:** `galacticus_tasks_evolve_tree`  
**Description:** Implements the task of evolving merger trees.  
**Code lines:** 313  
**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:** 230  
**Contained by:** module `galacticus_tasks_evolve_tree`  
**Modules used:** `galacticus_display`  
`galacticus_output_merger_tree`  
`galacticus_nodes`  
`galacticus_output_times`  
`input_parameters`  
`iso_varying_string`  
`memory_management`  
`merger_tree_mass_accretion_history`  
`merger_tree_output_structure`  
`merger_trees_evolve`  
`merger_trees_monotonic_mass_growth`  
`merger_trees_prune_branches`  
`merger_trees_prune_hierarchy`  
`merger_trees_regrid_times`  
`merger_trees_write`  
`string_handling`  
`system_load`

**subroutine:** `get_tree`  
**Description:** Get a tree to process.  
**Code lines:** 56  
**Contained by:** module `galacticus_tasks_evolve_tree`  
**Modules used:** `galacticus_display`  
`galacticus_output_merger_tree`  
`galacticus_meta_tree_timing`  
`galacticus_nodes`  
`merger_tree_construction`  
`node_component_black_hole_standard`  
`node_component_disk_exponential`  
`node_component_dark_matter_profile_scale`  
`node_component_disk_very_simple`  
`node_component_spheroid_standard`  

**file:** `geometry.coordinate_systems.F90`  
**Description:** Contains a module which implements calculations related to coordinate systems and transformations.  
**Code lines:** 103

**module:** `coordinate_systems`  
**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`  

781
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: halo_mass_functions.tasks.F90
Description: Contains a module which implements calculations of halo mass functions and related properties for output.
Code lines: 303

module: halo_mass_function_tasks
Description: Implements calculations of halo mass functions and related properties for output.
Code lines: 283
Contained by: file halo_mass_functions.tasks.F90
Modules used: io_hdf5
Used by: program halo_mass_functions

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: 175
Contained by: module halo_mass_function_tasks
Modules used: cosmology_functions critical_overdensity dark_matter_halo_biases dark_matter_halo_scales
18.1. Program units

**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:* 59
*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 stellar mass function.
*Code lines:* 334

**module: conditional_stellar_mass_functions_behroozi2010**
*Description:* Implements the Behroozi et al. [2010] fitting function descriptions of the conditional stellar mass function.
*Code lines:* 314
*Contained by:* file `halo_model.conditional_stellar_mass_function.Behroozi2010.F90` 
*Modules used:* `tables` 
*Used by:* subroutine `conditional_stellar_mass_functions_behroozi2010_initialize`

**subroutine: conditional_stellar_mass_functions_behroozi2010_initialize**
*Description:* Initializes the “Behroozi2010” conditional stellar mass function method.
*Code lines:* 161
*Contained by:* module `conditional_stellar_mass_functions_behroozi2010` 
*Modules used:* `input_parameters iso_varying_string`

**function: cumulative_conditional_stellar_mass_function_behroozi2010**
*Description:* Computes the cumulative conditional mass function, \( \langle N(M_*|M_{halo}) \rangle \equiv \phi(M_*|M_{halo}) \) using the fitting formula of Behroozi et al. [2010].
*Code lines:* 13
*Contained by:* module `conditional_stellar_mass_functions_behroozi2010`

**subroutine: cumulative_conditional_stellar_mass_function_compute**
*Description:* Computes the cumulative conditional mass function, \( \langle N(M_*|M_{halo}) \rangle \equiv \phi(M_*|M_{halo}) \) using the fitting formula of Behroozi et al. [2010].
*Code lines:* 62
*Contained by:* module `conditional_stellar_mass_functions_behroozi2010`
function: cumulative_conditional_stellar_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_stellar_mass_functions_behroozi2010

function: fshmrinverse
Description: The median stellar mass vs. halo mass relation functional form from Behroozi et al. [2010].
Code lines: 17
Contained by: module conditional_stellar_mass_functions_behroozi2010

file: halo_model.conditional_stellar_mass_function.F90
Description: Contains a module which implements empirical models of conditional stellar mass functions.
Code lines: 131

module: conditional_stellar_mass_functions
Description: Implements empirical models of conditional stellar mass functions.
Code lines: 111
Contained by: file halo_model.conditional_stellar_mass_function.F90
Modules used: iso_varying_string
Used by: function stellar_mass_function_integrand function xi_integrand

subroutine: conditional_stellar_mass_functions_initialize
Description: Initialize the conditional stellar mass function module.
Code lines: 52
Contained by: module conditional_stellar_mass_functions
Modules used: conditional_stellar_mass_functions_behroozi2010 galacticus_error input_parameters

function: cumulative_conditional_stellar_mass_function
Description: Returns the cumulative conditional stellar mass function at a stellar mass of massStellar in a halo of mass massHalo.
Code lines: 12
Contained by: module conditional_stellar_mass_functions

function: cumulative_conditional_stellar_mass_function_variance
Description: Returns the cumulative conditional stellar mass function at a stellar mass of massStellar in a halo of mass massHalo.
Code lines: 12
Contained by: module conditional_stellar_mass_functions

file: hot_halo.density_profile.F90
Description: Contains a module that implements calculations of the hot halo gas density profile.
Code lines: 282
module: hot_halo_density_profile

Description: Implements calculations of the hot halo gas density profile.

Code lines: 262

Contained by: file hot_halo_density_profile.F90

Modules used:
- galacticus_nodes
- hot_halo_density_profile_null
- iso_varying_string

Used by:
- function cooling_radius_isothermal
- function cooling_radius_root
- function cooling_rate_white_frenk
- function galactic_structure_density
- subroutine galactic_structure_radii_initial
- subroutine galactic_structure_rotation_curve
- subroutine galactic_structure_rotation_curve_gradient
- subroutine hot_halo_ram_pressure_force
- subroutine node_component_hot_halo_std_rate_compute

function: hot_halo_density

Description: Return the density of the hot halo in thisNode at radius radius.

Code lines: 13

Contained by: module hot_halo_density_profile

subroutine: hot_halo_density_initialize

Description: Initialize the hot halo density profile module.

Code lines: 42

Contained by: module hot_halo_density_profile

Modules used:
- galacticus_nodes
- input_parameters

function: hot_halo_density_log_slope

Description: Return the density of the hot halo in thisNode at radius radius.

Code lines: 13

Contained by: module hot_halo_density_profile

function: hot_halo_enclosed_mass

Description: Return the enclosed mass of the hot halo in thisNode at radius radius.

Code lines: 13

Contained by: module hot_halo_density_profile

function: hot_halo_profile_density_task

Description: Computes the density at a given position for a dark matter profile.

Code lines: 16
function: hot_halo_profile_enclosed_mass_task
Description: Computes the mass within a given radius for a dark matter profile.
Code lines: 18
Contained by: module hot_halo_density_profile
Modules used:

function: hot_halo_profile_radial_moment
Description: Returns a radial moment of the hot gas profile in thisNode to the specified radius.
Code lines: 11
Contained by: module hot_halo_density_profile

function: hot_halo_profile_rotation_curve_gradient_task
Description: Computes the rotation curve gradient at a given radius for the hot halo density profile.
Code lines: 24
Contained by: module hot_halo_density_profile
Modules used:
galactic_structure_options numerical_constants_math numerical_constants_physical

function: hot_halo_profile_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_density_profile
Modules used:
galactic_structure_options numerical_constants_physical

function: hot_halo_profile_rotation_normalization
Description: Returns the relation between specific angular momentum and rotation velocity (assuming a rotation velocity that is constant in radius) for thisNode. Specifically, the normalization, \( A \), returned is such that \( V_{\text{rot}} = AJ/M \).
Code lines: 13
Contained by: module hot_halo_density_profile

file: hot_halo_density_profile.cored_isothermal.F90
Description: Contains a module which implements a cored isothermal profile for hot gas halos.
Code lines: 235

module: hot_halo_density_profile.cored_isothermal
Description: Implements a cored isothermal profile for hot gas halos.
Code lines: 215
Contained by: file hot_halo_density_profile.cored_isothermal.F90
Used by: module hot_halo_density_profile

function: density_normalization_factor
Description: Computes the density profile normalization factor for a given core radius and outer radius.
Code lines: 25
Contained by: module hot_halo_density_profile.cored_isothermal
subroutine: hot_halo_density_cored_isothermal
Description: Initialize the cored isothermal hot halo density profile module.
Code lines: 22
Contained by: module hot_halo_density_profile_cored_isothermal
Modules used: galacticus_nodes iso_varying_string

function: hot_halo_density_cored_isothermal_enclosed_mass_get
Description: Compute the mass enclosed within radius radius in a cored isothermal hot halo density profile for thisNode.
Code lines: 38
Contained by: module hot_halo_density_profile_cored_isothermal
Modules used: galacticus_nodes hot_halo_density_cored_isothermal_core_radii

function: hot_halo_density_cored_isothermal_get
Description: Compute the density at radius radius in a cored isothermal hot halo density profile for thisNode.
Code lines: 18
Contained by: module hot_halo_density_profile_cored_isothermal
Modules used: galacticus_nodes hot_halo_density_cored_isothermal_core_radii numerical_constants_math

function: hot_halo_density_cored_isothermal_log_slope_get
Description: Compute the density at radius radius in a cored isothermal hot halo density profile for thisNode.
Code lines: 13
Contained by: module hot_halo_density_profile_cored_isothermal
Modules used: galacticus_nodes hot_halo_density_cored_isothermal_core_radii

function: hot_halo_profile_radial_moment_cored_isothermal_get
Description: Return the normalization of the rotation velocity vs. specific angular momentum relation.
Code lines: 45
Contained by: module hot_halo_density_profile_cored_isothermal
Modules used: galacticus_nodes hot_halo_density_cored_isothermal_core_radii numerical_comparison numerical_constants_math

function: hot_halo_profile_rotation_normalization_cored_isothermal_get
Description: Return the normalization of the rotation velocity vs. specific angular momentum relation.
Code lines: 26
Contained by: module hot_halo_density_profile_cored_isothermal
Modules used: galacticus_nodes hot_halo_density_cored_isothermal_core_radii

file: hot_halo.density_profile.cored_isothermal.core_radius.F90
Description: Contains a module which implements calculations of the core radius in cored isothermal hot halo profiles.

Code lines: 115

module: hot_halo_density_cored_isothermal_core_radii
Description: Implements calculations of the core radius in cored isothermal hot halo profiles.
Code lines: 95
Contained by: file hot_halo.density_profile.cored_isothermal.core_radius.F90

Modules used: galacticus_nodes iso_varying_string

Used by:
function hot_halo_density_cored_isothermal_enclosed_mass_get
function hot_halo_density_cored_isothermal_get
function hot_halo_density_cored_isothermal_log_slope_get
function hot_halo_profile_radial_moment_cored_isothermal_get
function hot_halo_profile_rotation_normalization_cored_isothermal_get

subroutine: hot_halo_density_cored_isothermal_core_radii_initialize
Description: Initialize the cored isothermal hot halo profile core radius module.
Code lines: 54
Contained by: module hot_halo_density_cored_isothermal_core_radii

Modules used: galacticus_error hot_halo_density_cored_isothermal_core_radii_growing_core

hot_halo_density_cored_isothermal_core_radii_growing_core
input_parameters

function: hot_halo_density_cored_isothermal_core_radius
Description: Returns the radius (in Mpc) of the core in a cored isothermal hot halo density profile for thisNode.
Code lines: 12
Contained by: module hot_halo_density_cored_isothermal_core_radii

file: hot_halo.density_profile.cored_isothermal.core_radius.growing_core.F90
Description: Contains a module which implements a calculation of the core radius in the hot halo density profile that is a fraction of the dark matter profile scale radius, but which grows as gas is depleted to keep the density at the virial radius constant (similar to the algorithm of [Cole et al., 2000]).
Code lines: 205

module: hot_halo_density_cored_isothermal_core_radii_growing_core
Description: Implements a calculation of the core radius in the hot halo density profile that is a fraction of the dark matter profile scale radius, but which grows as gas is depleted to keep the density at the virial radius constant (similar to the algorithm of [Cole et al., 2000]).
Code lines: 183
Contained by: file hot_halo.density_profile.cored_isothermal.core_radius.growing_core.F90

Modules used: galacticus_nodes tables

Used by:
subroutine galacticus_state_retrieve
subroutine galacticus_state_store
subroutine hot_halo_density_cored_isothermal_core_radii_initialize

function: growing_core_virial_density_function

788
18.1. Program units

**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_density_cored_isothermal_core_radii_growing_core

**subroutine:** hot_halo_density_cored_isothermal_core_radii_growing_core

**Description:** Initializes the “growing core” cored isothermal hot halo profile core radius module.

**Code lines:** 38

**Contained by:** module hot_halo_density_cored_isothermal_core_radii_growing_core

**Modules used:** galacticus_error input_parameters iso_varying_string

**subroutine:** hot_halo_density_cored_isothermal_core_radius_growing_core_state_retrieve

**Description:** Retrieve the tabulation state from the file.

**Code lines:** 12

**Contained by:** module hot_halo_density_cored_isothermal_core_radii_growing_core

**Modules used:** fgs1

**subroutine:** hot_halo_density_cored_isothermal_core_radius_growing_core_state_store

**Description:** Write the tabulation state to file.

**Code lines:** 9

**Contained by:** module hot_halo_density_cored_isothermal_core_radii_growing_core

**Modules used:** fgs1

**function:** hot_halo_density_cored_isothermal_core_radius_growing_core

**Description:** Returns the radius (in Mpc) of the core radius in a cored isothermal hot halo density profile. Assumes that the radius is a fraction of the dark matter profile scale radius, but which grows as gas is depleted to keep the density at the virial radius constant (similar to the algorithm of [Cole et al., 2000]).

**Code lines:** 72

**Contained by:** module hot_halo_density_cored_isothermal_core_radii_growing_core

**Modules used:** cosmological_parameters dark_matter_halo_scales

**file:** hot_halo_density_profile.cored_isothermal.core_radius.virial_radius_fraction.F90

**Description:** Contains a module which implements a calculation of the core radius in the hot halo density profile that is a fixed fraction of the virial radius.

**Code lines:** 74

**module:** hot_halo_density_cored_isothermal_core_radii_virial_fraction

**Description:** Implements a calculation of the core radius in the hot halo density profile that is a fixed fraction of the virial radius.

**Code lines:** 53

**Contained by:** file hot_halo_density_profile.cored_isothermal.core_radius.virial_radius_fraction.F90

**Used by:** subroutine hot_halo_density_cored_isothermal_core_radii_vf_initialize

**subroutine:** hot_halo_density_cored_isothermal_core_radii_vf_initialize

**Description:** Initializes the “virial radius fraction” cored isothermal hot halo profile core radius module.
18. Source Code Documentation

**Code lines:** 23  
**Contained by:** module *hot_halo_density_cored_isothermal_core_radii_virial_fraction*  
**Modules used:** *input_parameters* *iso_varying_string*

**function:** *hot_halo_density_cored_isothermal_core_radii_virial_fraction*  
**Description:** Returns the radius (in Mpc) of the core radius in a cored isothermal hot halo density profile. Assumes that the radius is a fixed fraction of the halo virial radius.  
**Code lines:** 11  
**Contained by:** module *hot_halo_density_cored_isothermal_core_radii_virial_fraction*  
**Modules used:** *dark_matter_halo_scales* *galacticus_nodes*

**file:** *hot_halo.density_profile.null.F90*  
**Description:** Contains a module which implements a null hot gas halo density profile.  
**Code lines:** 106

**module:** *hot_halo_density_profile_null*  
**Description:** Implements a null hot gas halo density profile.  
**Code lines:** 86  
**Contained by:** file *hot_halo.density_profile.null.F90*  
**Used by:** module *hot_halo_density_profile*

**subroutine:** *hot_halo_density_null*  
**Description:** Initialize the null hot halo density profile module.  
**Code lines:** 19  
**Contained by:** module *hot_halo_density_profile_null*  
**Modules used:** *iso_varying_string*

**function:** *hot_halo_density_null_enclosed_mass_get*  
**Description:** Compute the mass enclosed within radius *radius* in a null hot halo density profile for *thisNode*.  
**Code lines:** 9  
**Contained by:** module *hot_halo_density_profile_null*  
**Modules used:** *galacticus_nodes*

**function:** *hot_halo_density_null_get*  
**Description:** Compute the density at radius *radius* in a null hot halo density profile for *thisNode*.  
**Code lines:** 9  
**Contained by:** module *hot_halo_density_profile_null*  
**Modules used:** *galacticus_nodes*

**function:** *hot_halo_density_null_log_slope_get*  
**Description:** Compute the density at radius *radius* in a null hot halo density profile for *thisNode*.  
**Code lines:** 9  
**Contained by:** module *hot_halo_density_profile_null*  
**Modules used:** *galacticus_nodes*

**function:** *hot_halo_density_null_radial_moment_get*  
**Description:** Return a radial moment in a null hot halo density profile for *thisNode*.  
**Code lines:** 9
18.1. Program units

**Contained by:** module `hot_halo_density_profile_null`  
**Modules used:** `galacticus_nodes`

**function:** `hot_halo_density_null_rotation_normalization_get`  
**Description:** Return the normalization of the rotation velocity vs. specific angular momentum relation in a null hot halo density profile for `thisNode`.

**Code lines:** 8  
**Contained by:** module `hot_halo_density_profile_null`  
**Modules used:** `galacticus_nodes`

**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 `ram_pressure_stripping_mass_loss_rate_disk_simple`

**function:** `hot_halo_ram_pressure_force`  
**Description:** Return the ram pressure force for the hot halo of `thisNode`.

**Code lines:** 60  
**Contained by:** module `hot_halo_ram_pressure_forces`  
**Modules used:** `galacticus_error galacticus_nodes`  
`hot_halo_ram_pressure_force_font2008 hot_halo_ram_pressure_force_null`  
`input_parameters`

**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:** 83

**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:** 62  
**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`

**function:** `hot_halo_ram_pressure_force_font2008_get`  
**Description:** Computes the hot halo ram pressure force

**Code lines:** 28  
**Contained by:** module `hot_halo_ram_pressure_force_font2008`  
**Modules used:** `hot_halo_density_profile 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_standard_rate_compute`

### 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` `hot_halo_ram_pressure_stripping_virial_radii`
18.1. Program units

**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:** 144

**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:** 123

**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:** 52

**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:** 19

**Contained by:** module hot_halo_ram_pressure_stripping_font2008

**Modules used:** galactic_structure_enclosed_masses galactic_structure_options hot_halo_density_profile numerical_constants_physical

**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:** 56

**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:** 35

**Contained by:** file hot_halo.ram_pressure_stripping.virial_radius.F90

**Used by:** function hot_halo_ram_pressure_stripping_virial_radius.F90
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:* 10

*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**  **iso_varying_string**

*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*.
18.1. Program units

**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:** 70
**module:** hot_halo_temperature_profile_virial
**Description:** Implements an isothermal (virial temperature) profile for hot gas halos.

**Code lines:** 50
**Contained by:** file hot_halo.temperature_profile.virial.F90
**Used by:** module hot_halo_temperature_profile

**function:** hot_halo_temperature_logarithmic_slope_virial_get
**Description:** Compute the logarithmic slope of the temperature at radius $r$ in an isothermal temperature profile for thisNode.

**Code lines:** 10
**Contained by:** module hot_halo_temperature_profile_virial

**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 $r$ in an isothermal (virial) temperature profile for thisNode.

**Code lines:** 10
**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
function: derfc1
Code lines: 64
Contained by: module incomplete_gamma

function: derfi
Code lines: 97
Contained by: module incomplete_gamma

function: dgam1
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: dgmln1
Code lines: 13
Contained by: module incomplete_gamma

function: dlnrel
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 incompleteGamma

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
   Contained by: module incomplete_gamma

file: instruments.filters.F90
   Description: Contains a module which implements calculations of filter response curves.
   Code lines: 150

module: instruments_filters
   Description: Implements calculations of filter response curves.
   Code lines: 130
   Contained by: file instruments.filters.F90
   Modules used: fgsl iso_varying_string
   Used by: function filter_luminosity_integrand function filter_luminosity_integrand_ab
            function stellar_population_luminosity subroutine stellar_population_properties_luminosities_initialize

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_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.
18. Source Code Documentation

**Code lines:** 46  
**Contained by:** module `instruments_filters`  
**Modules used:**  
- `fox_dom`  
- `galacticus_error`  
- `galacticus_input_paths`  
- `io_xml`  
- `memory_management`  

**Type:** `filtertype`  
**Description:** A structure which holds filter response curves.  

**Code lines:** 9  
**Contained by:** module `instruments_filters`  

**File:** `intergalactic_medium.state.F90`  
**Description:** Contains a module that implements calculations of the intergalactic medium thermal and ionization state.  

**Code lines:** 296  
**Module:** `intergalactic_medium_state`  
**Description:** Implements calculations of the intergalactic medium thermal and ionization state.  

**Code lines:** 276  
**Contained by:** file `intergalactic_medium.state.F90`  
**Modules used:**  
- `tables`  

**Subroutines used:**  
- `accretion_halos_simple_initialize`  
- `galacticus_state_retrieve`  
- `galacticus_state_store`  

**Function:** `igm_state_electron_scattering_integrand`  
**Description:** Integrand for electron scattering optical depth calculations.  

**Code lines:** 21  
**Contained by:** module `intergalactic_medium_state`  
**Modules used:**  
- `cosmological_parameters`  
- `cosmology_functions`  
- `iso_c_binding`  
- `numerical_constants_astronomical`  
- `numerical_constants_physical`  

**Subroutine:** `igm_state_electron_scattering_tabulate`  
**Description:** Construct a table of electron scattering optical depth as a function of cosmological time.  

**Code lines:** 43  
**Contained by:** module `intergalactic_medium_state`  
**Modules used:**  
- `cosmology_functions`  
- `fgsl`  
- `iso_c_binding`  
- `numerical_integration`  

**Function:** `intergalactic_medium_electron_fraction`  
**Description:** Return the electron fraction in the intergalactic medium at the specified time.  

**Code lines:** 10  
**Contained by:** module `intergalactic_medium_state`  

**Function:** `intergalactic_medium_electron_scattering_optical_depth`  
**Description:** Return the electron scattering optical depth from the present day back to the given time in the intergalactic medium.  

**Code lines:** 24  
**Contained by:** module `intergalactic_medium_state`
18.1. Program units

**Modules used:** galacticus_error

**function:** intergalactic_medium_electron_scattering_time

*Description:* Return the cosmological time at which the given electron scattering opticalDepth is reached (integrating from the present day) in the intergalactic medium.

*Code lines:* 34

*Contained by:* module intergalactic_medium_state

**Modules used:** cosmology_functions galacticus_error

**subroutine:** intergalactic_medium_state_initialize

*Description:* Initialize the intergalactic medium state module.

*Code lines:* 57

*Contained by:* module intergalactic_medium_state

**Modules used:** galacticus_error input_parameters intergalactic_medium_state_file intergalactic_medium_state_recfast iso_varying_string

**subroutine:** intergalactic_medium_state_state_retrieve

*Description:* Retrieve the tabulation state from the file.

*Code lines:* 12

*Contained by:* module intergalactic_medium_state

**Modules used:** fgsl

**subroutine:** intergalactic_medium_state_state_store

*Description:* Write the tablulation state to file.

*Code lines:* 9

*Contained by:* module intergalactic_medium_state

**Modules used:** fgsl

**function:** intergalactic_medium_temperature

*Description:* Return the temperature of the intergalactic medium at the specified time.

*Code lines:* 10

*Contained by:* module intergalactic_medium_state

**file:** intergalatic_medium.state.RecFast.F90

*Description:* Contains a module that implements calculations of the intergalactic medium thermal and ionization state using RecFast.

*Code lines:* 100

**module:** intergalactic_medium_state_recfast

*Description:* Implements calculations of the intergalactic medium thermal and ionization state using RecFast.

*Code lines:* 78

*Contained by:* file intergalatic_medium.state.RecFast.F90

**Modules used:** iso_varying_string

**Used by:** subroutine intergalactic_medium_state-initialize

**subroutine:** intergalactic_medium_state_recfast_initialize
18. Source Code Documentation

**Description:** Initializes the “RecFast” intergalactic medium state module.

**Code lines:** 64

**Contained by:** module intergalactic_medium_state_recfast

**Modules used:**
- cosmological_parameters
- fox_wxml
- galacticus_input_paths
- input_parameters
- intergalactic_medium_state_file
- numerical_constants_astronomical
- system_command

**file:** intergalactic_medium.state.file.F90

**Description:** Contains a module that implements calculations of the intergalactic medium thermal and ionization state read from a file.

**Code lines:** 169

**module:** intergalactic_medium_state_file

**Description:** Implements calculations of the intergalactic medium thermal and ionization state read from a file.

**Code lines:** 147

**Contained by:** file intergalactic_medium.state.file.F90

**Modules used:**
- fgsl
- iso_varying_string

**Used by:**
- subroutine intergalactic_medium_state_recfast_initialize
- subroutine intergalactic_medium_state_initialize

**function:** intergalactic_medium_electron_fraction_file

**Description:** Return the electron fraction in the intergalactic medium at the specified time by interpolating in tabulated data.

**Code lines:** 13

**Contained by:** module intergalactic_medium_state_file

**Modules used:** numerical_interpolation

**subroutine:** intergalactic_medium_file_set_file

**Description:** Allow an external module to set the filename to be used for intergalactic medium state data.

**Code lines:** 6

**Contained by:** module intergalactic_medium_state_file

**function:** intergalactic_medium_state_file_current_file_format_version

**Description:** Return the current file format version of intergalactic medium state files.

**Code lines:** 6

**Contained by:** module intergalactic_medium_state_file

**subroutine:** intergalactic_medium_state_file_initialize

**Description:** Initializes the “file” intergalactic medium state module.

**Code lines:** 26

**Contained by:** module intergalactic_medium_state_file

**Modules used:** input_parameters

**subroutine:** intergalactic_medium_state_file_read_data

**Description:** Read in data describing the state of the intergalactic medium.

**Code lines:** 39

**Contained by:** module intergalactic_medium_state_file
18.1. Program units

**Modules used:**
- cosmology_functions
- fox_dom
- galacticus_error
- io_xml

**function:** intergalactic_medium_temperature_file

**Description:** Return the temperature of the intergalactic medium at the specified time by interpolating in tabulated data.

**Code lines:** 13

**Contained by:** module intergalactic_medium_state_file

**Modules used:**
- numerical_interpolation

**file:** iso_varying_string.F90

**Code lines:** 2674

**module:** iso_varying_string

**Code lines:** 2633

**Contained by:** file iso_varying_string.F90

**Used by:**
- program tests_excursion_sets
- program halo_mass_functions
- program power_spectra
- module accretion_halos

- subroutine accretion_halos_simple_initialize
- subroutine accretion_disks_eddington_initialize
- subroutine accretion_disks_shakura_sunyaev_initialize
- module atomic_cross_sections_ionization_photo
- subroutine atomic_data_initialize

- subroutine atomic_rate_ionization_collisional_verner_initialize
- subroutine atomic_rate_recombination_radiative_verner_initialize
- subroutine black_hole_binary_initial_radii

- subroutine black_hole_binary_initial_radii_volonteri_2003_initialize
- subroutine black_hole_binary_initial_radii_tidal_radius_initialize
- module black_hole_binary_recoil_velocities
- subroutine black_hole_binary_recoil_velocity_standard_initialize
- subroutine black_hole_binary_recoil_velocity_null_initialize
- subroutine black_hole_binary_recoil_velocity_null_initialize

**module:** accretion_disks

**subroutine:**
- subroutine accretion_disks_eddington_initialize
- subroutine accretion_disks_shakura_sunyaev_initialize
- subroutine atomic_rate_ionization_collisional_verner_initialize
- subroutine atomic_rate_recombination_radiative_verner_initialize
- subroutine black_hole_binary_recoil_velocity_standard_initialize
- subroutine black_hole_binary_recoil_velocity_null_initialize
- subroutine black_hole_binary_recoil_velocity_null_initialize
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
module cooling_functions_cmb_compton
subroutine cooling_function_not_matched
module cooling_functions_atomic_cie_cloudy
module cooling_functions_molecular_-hydrogen_galli_palla
subroutine cooling_radius_isothermal_-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 freefall_radius_dark_-matter_halo_initialize
subroutine freefall_time_available_-haloformation_initialize
subroutine infall_radius_cooling_-freefall_initialize
module cooling_specific-angular_-momenta
subroutine cooling_specific_am_mean_-initialize
subroutine cooling_time_available_wf_-initialize
module cosmology_functions
subroutine cosmology_functions_matter_-lambda_initialize
subroutine dark_matter_mass_accretion_-wechsler2002_initialize
module dark_matter_halos_mass_-loss_rates
module black_hole_binary_mergers
module chemical_reaction_rates
subroutine chemical_reaction_rates_-null_initialize
module chemical_states
module cooling_functions_cie_file
module cooling_functions
module cooling_functions_atomic_cie_cloudy
module cooling_functions_molecular_-hydrogen_galli_palla
subroutine cooling_radius_simple_-initialize
module cooling_rates
subroutine cooling_rate_white_frenk_-initialize
subroutine cooling_rate_simple_-scaling_initialize
module cooling_times
module freefall_radii
subroutine cooling_freefall_times_-available
module cooling_infall_radii
subroutine infall_radius_cooling_-radius_initialize
module cooling_specific_am_-constant_rotation_initialize
module cooling_times_available
subroutine cooling_time_available_wf_-initialize
module dark_matter_halos_mass_-loss_rate_null_initialize
subroutine dark_matter_halos_mass_loss_rate_vandenbosch_initialize
module halo_spin_distributions

subroutine halo_spin_distribution_lognormal_initialize
module dark_matter_profiles

subroutine dark_matter_profile_isothermal_initialize
subroutine dark_matter_concentrations_gao2008_initialize
subroutine dark_matter_concentrations_nfw1996_initialize
subroutine dark_matter_concentrations_zhao2009_initialize
subroutine dark_matter_shapes_gao2008_initialize
subroutine events_node_merger_do_slh

function node_subhalo_promotion
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_radii_radii_adiabatic_initialize
subroutine galactic_structure_radii_radii_linear_initialize
module galacticus_display

module galacticus_input_paths
subroutine galacticus_time_per_tree_file_initialize
subroutine galacticus_meta_evolver_profiler_output
subroutine galacticus_build_output
module galacticus_merger_tree_output_filters

subroutine halo_spin_distribution_bett2007_initialize
subroutine halo_spin_distribution_delta_function_initialize
subroutine dark_matter_profile_einasto_initialize
subroutine dark_matter_profile_nfw_initialize
module dark_matter_profiles
subroutine dark_matter_profiles_concentrations
subroutine dark_matter_concentrations_munozcuartas2011_initialize
subroutine dark_matter_concentrations_prada2011_initialize
module dark_matter_profiles_shapes

function node_branch_jump
subroutine events_node_merger_initialize_slh
subroutine galactic_dynamics_bar_instabilities_eln_initialize
subroutine galactic_dynamics_bar_instabilities_null_initialize
module galactic_structure_radii
subroutine solve_for_radius
module galactic_structure_initial_radii
subroutine galactic_structure_initial_radii_static_initialize
subroutine galactic_structure_radii_simple_initialize
subroutine galacticus_error_report_varstr
module galacticus_meta_compute_times
subroutine galacticus_meta_evolver_profile
module galacticus_output_open
module galacticus_output_merger_tree
subroutine galacticus_merger_tree_output_filter_lightcone_initialize
subroutine galacticus_merger_tree_output_filter_luminosity_initialize
subroutine galacticus_output_tree_half_light_names
module galacticus_output_trees_rotation_curve
subroutine starformation_histories_metallicity_split_initialize
subroutine starformation_histories_null_initialize
function galacticus_version
module galacticus_state
subroutine halo_mass_state_open_file
module conditional_stellar_mass_functions
subroutine hot_halo_density_cored_isothermal
subroutine hot_halo_density_cored_isothermal_core_radii_gc_initialize
subroutine hot_halo_density_null
subroutine hot_halo_ram_pressure_force_font2008_initialize
module hot_halo_ram_pressure_stripping
subroutine hot_halo_ram_pressure_stripping_virial_radii_initialize
subroutine hot_halo_temperature_virial
subroutine intergalactic_medium_state_initialize
module intergalactic_medium_state_file
module merger_tree_branching
subroutine generalized_press_schechter_branching_initialize
subroutine modified_press_schechter_branching_initialize
subroutine merger_tree_branching_modifiers_parkinson_initialize
module merger_tree_construction
module merger_tree_build
subroutine galacticus_merger_tree_output_filter_stellar_mass_initialize
subroutine galacticus_extra_output_halo_fourier_profile
module galacticus_output_star_formation_histories
subroutine starformation_history_metallicity_split
module galacticus_output_trees_velocity_dispersion
subroutine galacticus_version_output
function galacticus_task_evolve_tree
subroutine conditional_stellar_mass_functions_behroozi2010_initialize
module hot_halo_density_profile
module hot_halo_density_cored_isothermal_core_radii
subroutine hot_halo_density_cored_isothermal_core_radii_vf_initialize
module hot_halo_ram_pressure_forces
subroutine hot_halo_ram_pressure_force_null_initialize
module hot_halo_ram_pressure_stripping
module hot_halo_ram_pressure_stripping_virial_radii_initialize
module hot_halo_temperature_profile
module instruments_filters
module intergalactic_medium_state_file
function inverse_gamma_function_incomplete_complementary
function generalized_press_schechter_branch_mass
function generalized_press_schechter_subresolution_fraction
module merger_tree_branching_modifiers
subroutine merger_tree_branching_modifiers_null_initialize
subroutine merger_tree_build_cole2000_initialize
module merger_trees_construct_fully_specified
module merger_trees_mass_function_sampling
subroutine merger_trees_mass_function_sampling_halo_mf_initialize
subroutine merger_trees_mass_function_sampling_stellar_mf_initialize
module merger_tree_smooth_accretion
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_millennium_process
subroutine merger_tree_mass_accretion_history_output
module merger_trees_regrid_times
subroutine odeiv2_solve
function interpolate
function interpolate_derivative
function search_array_varstring
module chemical_abundances_structure
subroutine history_dump
subroutine kepler_orbits_dump
subroutine merger_tree_dump
subroutine basiccreatelinked
subroutine darkmatterprofilecreatelinked
subroutine formationtimecreatelinked
subroutine hothalocreatelinked
subroutine interoutputcreatelinked
subroutine node_component_basic_dump
subroutine node_component_basicnamingevolving_name_from_index
subroutine node_component_basicnull_name_from_index
subroutine node_component_basicstandard_name_from_index
subroutine node_component_blackhole_dump
subroutine node_component_blackholenull_dump
subroutine node_component_blackholesimple_dump
subroutine node_component_blackholenull_name_from_index
subroutine node_component_blackholesimple_name_from_index
subroutine merger_trees_mass_function_sampling_gaussian_initialize
subroutine merger_trees_mass_function_sampling_power_law_initialize
module merger_tree_read
module merger_trees_state_store
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_trees_simple_process
subroutine merger_tree_structure_output
module merger_trees_write
function interpolate
function root_finder_find
module abundances_structure
module chemical_structures
subroutine history_extend
module merger_tree_data_structure
module galacticus_nodes
subroutine history_dump
subroutine kepler_orbits_dump
subroutine merger_tree_dump
subroutine basiccreatelinked
subroutine darkmatterprofilecreatelinked
subroutine formationtimecreatelinked
subroutine hothalocreatelinked
subroutine interoutputcreatelinked
subroutine node_component_basic_dump
subroutine node_component_basicnamingevolving_name_from_index
subroutine node_component_basicnull_name_from_index
subroutine node_component_basicstandard_name_from_index
subroutine node_component_blackhole_dump
subroutine node_component_blackholenull_dump
subroutine node_component_blackholesimple_dump
subroutine node_component_blackholenull_name_from_index
subroutine node_component_blackholesimple_name_from_index
subroutine node_component_-blackholestandard_dump
subroutine node_component_-darkmatterprofile_dump
subroutine node_component_-darkmatterprofilenull_name_from_index
subroutine node_component_-darkmatterprofilescale_dump
subroutine node_component_-darkmatterprofilescale_name_from_index
subroutine node_component_-darkmatterprofileshapeshape_dump
subroutine node_component_-darkmatterprofileshapeshape_name_from_index
subroutine node_component_-disk_dump
subroutine node_component_-diskexponential_dump
subroutine node_component_-diskexponential_name_from_index
subroutine node_component_-disknull_dump
subroutine node_component_-formationtime_dump
subroutine node_component_-formationtimecole2000_dump
subroutine node_component_-formationtimecole2000_name_from_index
subroutine node_component_-formationtimenull_dump
subroutine node_component_-formationtimenull_name_from_index
subroutine node_component_-hothalo_dump
subroutine node_component_-hothalonull_dump
subroutine node_component_-hothalonull_name_from_index
subroutine node_component_-hothalostandard_dump
subroutine node_component_-hothaloverysimpleDump
subroutine node_component_-indices_dump
subroutine node_component_-indicesnull_dump
subroutine node_component_-indicesnull_name_from_index
subroutine node_component_-interoutput_dump
subroutine node_component_-interoutputnull_dump
subroutine node_component_-interoutputnull_name_from_index
subroutine node_component_-mergingstatistics_dump
subroutine node_component_-mergingstatisticsnull_dump
subroutine node_component_-mergingstatisticsnull_name_from_index
18.1. Program units

subroutine node_component_-mergingstatisticsrecent_name_from_index
subroutine node_component_-mergingstatisticsstandard_name_from_index
subroutine node_component_position_dump
subroutine node_component_positionnull_dump
subroutine node_component_positionpreset_dump
subroutine node_component_satellite_dump
subroutine node_component_satellitenull_dump
subroutine node_component_satellitepreset_dump
subroutine node_component_satellitestandard_dump
subroutine node_component_satelliteverysimple_dump
subroutine node_component_spheroid_dump
subroutine node_component_spheroidnull_dump
subroutine node_component_spheroidstandard_dump
subroutine node_component_spinnull_dump
subroutine node_component_spinpreset_dump
subroutine node_component_spinrandom_dump
subroutine node_dump
subroutine positioncreatelinked
subroutine spheroidcreatelinked
subroutine node_component_black_hole_-standard_output_properties
module node_component_disk_very_simple
subroutine node_component_merging_-statistics_recent_initialize
subroutine node_component_spheroid_-standard_post_evolve
subroutine radiation_initialize_-intergalactic_background
subroutine node_component_-mergingstatisticsstandard_dump
subroutine node_component_name_from_index
subroutine node_component_positionnull_dump
subroutine node_component_positionpreset_dump
subroutine node_component_satellite_-dump
subroutine node_component_satellitenull_name_from_index
subroutine node_component_satellitepreset_name_from_index
subroutine node_component_satellitestandard_name_from_index
subroutine node_component_satelliteverysimple_name_from_index
subroutine node_component_spheroidnull_dump
subroutine node_component_spheroidpreset_name_from_index
subroutine node_component_spheroidstandard_name_from_index
subroutine node_component_spinnull_name_from_index
subroutine node_component_spinpreset_name_from_index
subroutine node_component_spinrandom_name_from_index
function node_property_name_from_index
subroutine radiation_igb_file_initialize
function ram_pressure_stripping_mass_loss_rate_disk
routine ram_pressure_stripping_mass_loss_rate_disks_simple_init
routine satellite_time_until_merging_jiang2008_initialize
routine satellite_time_until_merging_lacey_cole2005_initialize
routine satellite_time_until_merging_lacey_cole_tormen_initialize
routine satellite_time_until_merging_wetzel_white_initialize
routine satellite_merging_mass_movements_baugh2005_initialize
routine satellite_merging_mass_movements_simple_initialize
routine satellite_time_until_merging_preset_initialize
routine satellite_merging_mass_movements_very_simple_initialize
routine satellite_merging_mass_movements_null_init
routine satellite_merging_mass_movements_standard_init
module satellite_merging_mass_movements
routine satellite_merging_mass_movements_progenitor_properties_cole2000_init
module satellite_merging_mass_movements_progenitors
routine satellite_merging_mass_movements_super_simple_initialize
routine satellite_time_until_merging_boylankolchin2008_initialize
routine satellite_time_until_merging_lacey_cole_initialize
routine satellite_time_until_merging_wetzel_white_initialize
module satellite_merging_mass_movements
routine satellite_merging_mass_movements_progenitor_properties_cole2000_init
module satellite_merging_mass_movements_progenitors
routine satellite_merging_mass_movements_super_simple_initialize
routine satellite_time_until_merging_boylankolchin2008_initialize
routine satellite_time_until_merging_lacey_cole_initialize
routine satellite_time_until_merging_wetzel_white_initialize
module satellite_merging_mass_movements
routine satellite_merging_mass_movements_progenitor_properties_cole2000_init
module satellite_merging_mass_movements_progenitors
routine satellite_merging_mass_movements_super_simple_initialize
routine satellite_time_until_merging_boylankolchin2008_initialize
routine satellite_time_until_merging_lacey_cole_initialize
routine satellite_time_until_merging_wetzel_white_initialize
module satellite_merging_mass_movements
routine satellite_merging_mass_movements_progenitor_properties_cole2000_init
module satellite_merging_mass_movements_progenitors
routine satellite_merging_mass_movements_super_simple_initialize
routine satellite_time_until_merging_boylankolchin2008_initialize
routine satellite_time_until_merging_lacey_cole_initialize
routine satellite_time_until_merging_wetzel_white_initialize
module satellite_merging_mass_movements
routine satellite_merging_mass_movements_progenitor_properties_cole2000_init
module satellite_merging_mass_movements_progenitors
routine satellite_merging_mass_movements_super_simple_initialize
routine satellite_time_until_merging_boylankolchin2008_initialize
routine satellite_time_until_merging_lacey_cole_initialize
routine satellite_time_until_merging_wetzel_white_initialize
module satellite_merging_mass_movements
routine satellite_merging_mass_movements_progenitor_properties_cole2000_init
module satellite_merging_mass_movements_progenitors
routine satellite_merging_mass_movements_super_simple_initialize
routine satellite_time_until_merging_boylankolchin2008_initialize
routine satellite_time_until_merging_lacey_cole_initialize
routine satellite_time_until_merging_wetzel_white_initialize
module satellite_merging_mass_movements
routine satellite_merging_mass_movements_progenitor_properties_cole2000_init
module satellite_merging_mass_movements_progenitors
routine satellite_merging_mass_movements_super_simple_initialize
routine satellite_time_until_merging_boylankolchin2008_initialize
routine satellite_time_until_merging_lacey_cole_initialize
routine satellite_time_until_merging_wetzel_white_initialize
module satellite_merging_mass_movements
routine satellite_merging_mass_movements_progenitor_properties_cole2000_init
module satellite_merging_mass_movements_progenitors
routine satellite_merging_mass_movements_super_simple_initialize
routine satellite_time_until_merging_boylankolchin2008_initialize
routine satellite_time_until_merging_lacey_cole_initialize
routine satellite_time_until_merging_wetzel_white_initialize
module satellite_merger_mass_movements
routine satellite_merger_mass_movements_progenitor_properties_cole2000_init
module satellite_merger_mass_movements_progenitors
routine satellite_merger_mass_movements_super_simple_initialize
routine satellite_sequence_until_merging_boylankolchin2008_finalize
routine satellite_sequence_until_merging_lacey_cole_finalize
routine satellite_sequence_until_merging_wetzel_white_finalize
module satellite_sequence_until_merging
routine satellite_sequence_until_merging_progenitor_properties_cole2000_finalize
module satellite_sequence_until_merging_progenitors
routine satellite_sequence_until_merging_progenitor_properties_standard_finalize
routine satellite_sequence_until_merging_progenitor_properties_super_finalize
routine virial_orbital_parameters_benson2005_initialize
routine virial_orbital_parameters_wetzel2010_finalize
routine virial_orbital_parameters_fixed_finalize
module virial_orbital_parameters
routine star_formation_imf_register_name_baugh2005_topheavy_finalize
routine star_formation_imf_register_name_chabrier_finalize
routine star_formation_imf_register_name_kennicutt_finalize
routine star_formation_imf_register_name_kroupa_finalize
routine star_formation_imf_register_name_millerscalo_finalize
routine star_formation_imf_register_name_salpeter_finalize
routine star_formation_imf_register_name_piecewisepowerlaw_finalize
routine imf_select_fixed_initialize
routine imf_select_disk_spheroid_powerlaw_finalize
routine imf_select_disk_spheroid_schaller_finalize
routine imf_select_disk_spheroid_salpeter_finalize
routine star_formation_feedback_disks_creasey2012_finalize
routine star_formation_feedback_disks_fixed_finalize
routine star_formation_feedback_disks_power_law_finalize
routine star_formation_feedback_disks_halo_scaling_finalize
routine star_formation_feedback_spheroids_creasey2012_finalize
routine star_formation_feedback_spheroids_fixed_finalize
routine star_formation_feedback_spheroids_power_law_finalize
routine star_formation_feedback_spheroids_halo_scaling_finalize
module star_formation_feedback_disks
routine star_formation_feedback_disks_creasey2012_finalize
routine star_formation_feedback_disks_fixed_finalize
routine star_formation_feedback_disks_power_law_finalize
routine star_formation_feedback_disks_halo_scaling_finalize
module star_formation_feedback_spheroids
routine star_formation_feedback_spheroids_creasey2012_finalize
routine star_formation_feedback_spheroids_fixed_finalize
routine star_formation_feedback_spheroids_power_law_finalize
routine star_formation_feedback_spheroids_halo_scaling_finalize
subroutine star_formation_feedback_spheroids_power_law_initialize
subroutine star_formation_expulsive_feedback_disks_null_initialize
module star_formation_feedback_expulsion_spheroids
subroutine star_formation_expulsive_feedback_spheroids_sw_initialize
module star_formation_rate_surface_density_disks
subroutine star_formation_rate_surface_density_disks_br_initialize
module 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
module star_formation_timescales_disks
subroutine star_formation_timescale_disks_dynamical_time_initialize
subroutine star_formation_timescale_disks_fixed_initialize
subroutine star_formation_timescale_disks_halo_scaling_initialize
subroutine star_formation_timescale_disks_integrated_sd_initialize
module star_formation_timescales_spheroids
subroutine star_formation_timescale_spheroids_dynamical_time_initialize
module stellar_feedback
subroutine stellar_feedback_standard_initialize
module supernovae_population_iii
subroutine supernovae_population_iii_hegerwoosley_initialize
subroutine supernovae_type_ia_nagashima_initialize
module stellar_tracks_initialize_file
subroutine stellar_tracks_initialize_file
module stellar_winds_leitherer1992_initialize
module stellar_population_properties
module stellar_population_properties_luminosities
module stellar_population_spectra_conroy
module stellar_population_spectra_file
module stellar_population_spectra_postprocess_lycSuppress
module stellar_population_spectra_postprocessing_madau1995
module stellar_population_spectra_postprocessing
module stellar_population_spectra_postprocessing
module stellar_population_spectra_postprocessing
module stellar_population_spectra_postprocessing_madau1995
module stellar_population_spectra_postprocessing_meiksin2006
subroutine stellar_population_spectra_postprocess_null_initialize
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_mass_scaling_null_initialize
subroutine excursion_sets_barriers_critical_overdensity_mass_scaling_wdm_initialize
subroutine excursion_sets_barriers_critical_overdensity_mass_linear_initialize
subroutine excursion_sets_barriers_critical_overdensity_mass_remap_null_initialize
module excursion_sets_barriers
subroutine excursion_sets_first_crossing_farahi
subroutine excursion_sets_first_crossing_zhang_hui_initialize
subroutine excursion_sets_first_crossing_zhang_hui_high_initialize
subroutine excursion_sets_first_crossing_linear_barrier_initialize
module dark_matter_halo_biases
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_sheth_tormen_initialize
subroutine halo_mass_function_tinker2008_initialize
subroutine growth_factor_simple_initialize
function power_spectrum_nonlinear_cosmicemu
module power_spectra
subroutine power_spectrum_nonlinear_linear_initialize
subroutine primordial_power_spectrum_power_law_initialize
subroutine power_spectrum_window_functions_sharp_kspace_initialize
subroutine power_spectrum_window_functions_top_hat_initialize
subroutine power_spectrum_window_functions_th_kss_hybrid_initialize
module spherical_collapse_matter_dark_energy
18.1. Program units

```plaintext
subroutine spherical_collape_delta_critical_initialize
module transfer_function_bbks
module transfer_function_eisenstein_hu
module transfer_functions_file

subroutine virial_density_bryan_norman_initialize
subroutine virial_density_kitayama_suto1996_initialize
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_ed

program tests_comoving_distance_open
program tests_cosmic_age_cosmological_constant
program tests_cosmic_age_dark_energy_cosmological_constant
program tests_cosmic_age_open

program test_hashes Cryptographic
program tests_linear_growth_ed
program tests_linear_growth_open

program test_nodes
program tests_power_spectrum
program tests_sigma

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_half
program tests_spherical_collapse_dark_energy_lambda
program tests_spherical_collapse_flat
program test_string_utilities
subroutine get_argument_varying_string
function formatted_date_and_time
module hashes

subroutine spherical_collape_delta_critical_initialize
module transfer_function_cmbfast
module transfer_functions
subroutine transfer_function_null_initialize
module virial_density_contrast

subroutine virial_density_fixed_initialize
program tests_io_hdf5
program test_prada2011_concentration

program test_zhao2009_flat
program test_zhao2009_open
program test_array_monotonicity
program tests_comoving_distance_ed

program tests_comoving_distance_open
program tests_cosmic_age_cosmological_constant
program tests_cosmic_age_dark_energy_cosmological_constant
program tests_cosmic_age_open

program test_hashes Cryptographic
program tests_linear_growth_ed
program tests_linear_growth_open

program test_nodes
program tests_power_spectrum
program tests_sigma

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_half
program tests_spherical_collapse_dark_energy_lambda
program tests_spherical_collapse_flat
program test_string_utilities
subroutine get_argument_varying_string
function formatted_date_and_time
module hashes
```

811
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_int8_1d
subroutine alloc_array_integer_int8_2d
subroutine alloc_array_logical_1d
subroutine alloc_array_logical_2d
subroutine code_memory_usage
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_int8_1d
subroutine dealloc_array_integer_int8_2d
subroutine dealloc_array_logical_1d
subroutine dealloc_array_logical_2d
subroutine dealloc_array_integer_int8_1d
subroutine dealloc_array_integer_int8_2d
subroutine dealloc_array_integer_int8_1d
subroutine dealloc_array_logical_1d
subroutine dealloc_array_real_kind_quad_1d
subroutine dealloc_array_real_kind_quad_1d
subroutine dealloc_array_real_kind_quad_1d
subroutine dealloc_array_real_kind_quad_1d
subroutine dealloc_array_real_kind_quad_1d
subroutine dealloc_array_real_kind_quad_1d
subroutine dealloc_array_real_kind_quad_1d
subroutine dealloc_array_real_kind_quad_1d
module string_handling
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

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
18.1. Program units

**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  
**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
Contained by: module iso_varying_string

interface: lle
Code lines: 4
Contained by: module iso_varying_string

function: lle_ch_vs
Code lines: 15
Contained by: module iso_varying_string

function: lle_vs_ch
Code lines: 15
Contained by: module iso_varying_string

function: lle_vs_vs
Code lines: 14
Contained by: module iso_varying_string

interface: llt
Code lines: 4
Contained by: module iso_varying_string

function: llt_ch_vs
Code lines: 15
Contained by: module iso_varying_string

function: llt_vs_ch
Code lines: 15
Contained by: module iso_varying_string

function: llt_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_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_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_ch
  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_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_target
Code lines: 19
Contained by: module iso_varying_string

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

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: fgsl
Used by: function node_component_disk_- exponential_rotation_curve_bessel_- exponential_rttn_crv_grdnt_bssl_fctrs
factors
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.
18. Source Code Documentation

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 \( x \).

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`

**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 `dark_matter_profile_kspace_nfw`
- function `dark_matter_profile_kspace_isothermal`
- program `test_math_special_functions`

**function: cosine_integral**

*Description:* Evaluate the \( \text{Ci}(x) \equiv \int_0^x dt \cos(t)/t \) cosine integral.

Code lines: 7
Contained by: module `exponential_integrals`
function: sine_integral
Description: Evaluate the Si(x) = \int_0^x dt \sin(t)/t sine integral.
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 stellar_population_spectra_-postprocess_meiksin2006

function: factorial
Description: Computes the factorial of argument.
Code lines: 8
Contained by: module factorials
Modules used: fgsl

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 dark_matter_profile_rotation_-normalization_einasto
function density_einasto_scale_free function enclosed_mass_einasto_scale_-free
function potential_einasto_scale_free subroutine radius_from_specific_-angular_momentum_table_make
subroutine stellar_population_spectra_-postprocess_meiksin2006

function: gamma_function
Description: Computes the gamma function.
Code lines: 7
18. Source Code Documentation

**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* `circularity_cumulative_-probability`

*subroutine* `virial_orbital_parameters_-`

*program* `test_math_special_functions wetzel2010_initialize`

**function: hypergeometric_1f1**

*Description:* Evaluate the \( _1F_1(a_1;b_1;x) \) hypergeometric function.

*Code lines:* 7
18.1. Program units

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.vector.F90
Description: Contains a module which implements calculations of vectors.
Code lines: 49

module: vectors
Description: Implements calculations of vectors.
Code lines: 29
Contained by: file math.vector.F90
Used by: subroutine galacticus_merger_tree_- output_filter_lightcone_initialize subroutine scan_for_mergers subroutine read_node_data program test_vectors

function: vector_magnitude
Description: Computes the magnitude of \( \text{vector}_1 \).
Code lines: 7
Contained by: module vectors

function: vector_product
Description: Computes the vector product of \( \text{vector}_1 \) and \( \text{vector}_2 \).
Code lines: 10
Contained by: module vectors

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
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
input_parameters
generalized_press_schechter_branching
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: 386

module: generalized_press_schechter_branching
  Description: Implements calculations of branching probabilities in generalized Press-Schechter theory.
  Code lines: 366
  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: 18
  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
18. Source Code Documentation

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

function: generalized_press_schechter_branching_maximum_step
Description: Return the maximum allowed step in $\delta_{\text{crit}}$ that a halo of mass \texttt{haloMass} at time $\texttt{deltaCritical}$ 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 \texttt{haloMass} at time $\texttt{deltaCritical}$ will undergo a branching to progenitors with mass greater than \texttt{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 \texttt{massResolution}, per unit change in $\delta_{\text{crit}}$ for a halo of mass \texttt{haloMass} at time $\texttt{deltaCritical}$. 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

$$\frac{d^2f}{d\ln M_{\text{child}}d\delta_{\text{c}}} = 2\sigma^2(M_{\text{child}}) \left. \frac{d\ln \sigma}{d\ln M} \right|_{M=M_{\text{child}}} \frac{dt}{d\delta_{\text{c}}} \frac{df_{12}}{dt},$$  \hspace{1cm} (18.8)

where $df_{12}/dt$ is the excursion set barrier crossing probability per unit time for the effective barrier $B'(S_{\text{child}}|S_{\text{parent}}, t) = B(S_{\text{child}}, t - \delta t) - B(S_{\text{parent}}, t)$ 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: 301

module: modified_press_schechter_branching
Description: Implements calculations of branching probabilities in modified Press-Schechter theory.

Code lines: 281

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
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: 11

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, `probability`. Typically, `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 `haloMass` at time `deltaCritical` should be allowed to take.
**Code lines:** 17
**Contained by:** module `modified_press_schechter_branching`

**function:** `modified_press_schechter_branching_probability`
**Description:** Return the probability per unit change in $\delta_{crit}$ that a halo of mass `haloMass` at time `deltaCritical` will undergo a branching to progenitors with mass greater than `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 `massResolution`, per unit change in $\delta_{crit}$ for a halo of mass `haloMass` at time `deltaCritical`. 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`
18.1. Program units

**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 merging_rate

**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 branching 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_parkinson.initialize

**function:** merger_tree_branching_modifier_parkinson

**Description:** Returns a modifier for merger tree branching rates using the Parkinson et al. [2008] algorithm.

**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
18. Source Code Documentation

*Description:* Contains a module which implements calculations of branching probabilities 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.construct.F90**

*Description:* Contains a module which constructs/destructs merger trees.

*Code lines:* 128

**module: merger_tree_construction**

*Description:* Constructs/destructs merger trees.

*Code lines:* 108

*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:* 81

*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


*Description:* Contains a module which implements building of merger trees using the algorithm of Cole et al. [2000].

*Code lines:* 295

**module: merger_tree_build_cole2000**

834
18.1. Program units

**Description:** Implements building of merger trees using the algorithm of Cole et al. [2000].

**Code lines:** 275

**Contained by:** file `merger_trees.construct.build.Cole2000.F90`

**Modules used:** `fgs1`

**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:** 63

**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:** 7

**Contained by:** module `merger_tree_build_cole2000`

**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:** 146

**Contained by:** module `merger_tree_build_cole2000`

**Modules used:**
- `critical_overdensity`
- `galacticus_nodes`
- `kind_numbers`
- `merger_tree_branching`
- `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:** 404

**module:** `merger_tree_build`

**Description:** Implements building of merger trees after drawing masses at random from a mass function.

**Code lines:** 384

**Contained by:** file `merger_trees.construct.build.F90`

**Modules used:**
- `galacticus_nodes`
- `iso_varying_string`
18. Source Code Documentation

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: iso_c_binding

subroutine: merger_tree_build_do
Description: Build a merger tree.
Code lines: 55
Contained by: module merger_tree_build
Modules used: galacticus_state

subroutine: merger_tree_build_initialize
Description: Initializes the merger tree building module.
Code lines: 276
Contained by: module merger_tree_build
Modules used: cosmology_functions

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

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

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

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: program optimal_sampling_smf 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_-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_initialze

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
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: 51

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: 31
Contained by: file merger_trees.construct.mass_function_sampling.halo_mass_function.F90
Used by: subroutine merger_trees_mass_function_sampling_initialze

function: merger_tree_construct_mass_function_sampling_halo_mf
Description: Computes the halo mass function sampling rate using a volume-limited sampling.
Code lines: 8
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: 9
Contained by: module merger_trees_mass_function_sampling_halo_mf
Modules used: 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

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
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_stellar_mf

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: fgs1 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
function: xi_integrand
Description: The integrand appearing in the $\xi$ function.
Code lines: 24
Contained by: module merger_trees_mass_function_sampling_stellar_mf
Modules used: conditional_stellar_mass_functions iso_c_binding

file: merger_trees.construct.read.F90
Description: Contains a module which implements reading of merger trees from an HDF5 file.
Code lines: 2533

module: merger_tree_read
Description: Implements reading of merger trees from an HDF5 file.
Code lines: 2513
Contained by: file merger_trees.construct.read.F90
Modules used: galacticus_nodes hdf5 io_hdf5 iso_varying_string kind_numbers
Used by: function merger_tree_create

subroutine: assign_isolated_node_indices
Description: Assign to each node the number of the corresponding isolated node.
Code lines: 42
Contained by: module merger_tree_read

subroutine: assign_mergers
Description: Assign pointers to merge targets.
Code lines: 42
Contained by: module merger_tree_read
Modules used: galacticus_error string_handling

subroutine: assign_scale_radii
Description: Assign scale radii to nodes.
Code lines: 68
Contained by: module merger_tree_read
Modules used: dark_matter_halo_scales galacticus_display galacticus_error root_finder

subroutine: assign_spin_parameters
Description: Assign spin parameters to nodes.
Code lines: 27
Contained by: module merger_tree_read
Modules used: dark_matter_profiles 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: build_child_and_sibling_links
Description: Build child and sibling links between nodes.
Code lines: 62
Contained by: module merger_tree_read
Modules used: memory_management

subroutine: build_descendent_pointers
Description: Builds pointers from each node to its descendant node.
Code lines: 44
Contained by: module merger_tree_read
Modules used: galacticus_display galacticus_error string_handling

subroutine: build_isolated_parent_pointers
Description: Create parent pointer links between isolated nodes and assign times and masses to those nodes.
Code lines: 81
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: 157
Contained by: module merger_tree_read
Modules used: cosmology_functions galacticus_error histories string_handling

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: 62
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: 35
18. Source Code Documentation

**Contained by:** module merger_tree_read

**Modules used:**
- galacticus_display
- memory_management
- sort
- string_handling

**function:** descendent_node_sort_index

**Description:** Return the sort index of the given descendentIndex.

**Code lines:** 8

**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:** 86

**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:** 11

**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:** 48

**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:** 220

**Contained by:** module merger_tree_read

**Modules used:** cosmology_functions galacticus_error
18.1. Program units

**subroutine: merger_tree_read_initialize**

*Description:* Initializes the merger tree reading module.
*Code lines:* 525
*Contained by:* module merger_tree_read
*Modules used:* cosmological_parameters, galacticus_error, input_parameters, numerical_comparison, power_spectra

describe subroutine

**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

describe function

**type: nodedata**

*Description:* Structure used to store raw data read from merger tree files.
*Code lines:* 8
*Contained by:* module merger_tree_read

describe type

**subroutine: read_node_data**

*Description:* Read node data from an HDF5 file.
*Code lines:* 98
*Contained by:* module merger_tree_read
*Modules used:* arrays_search, cosmology_functions, galacticus_error, memory_management, numerical_comparison, vectors

describe subroutine

**subroutine: read_particle_data**

*Description:* Read data on particle positions/velocities.
*Code lines:* 45
*Contained by:* module merger_tree_read
*Modules used:* cosmology_functions

describe subroutine

**subroutine: scan_for_branch_jumps**

*Description:* Search for subhalos which move between branches/trees.
*Code lines:* 122
*Contained by:* module merger_tree_read

describe subroutine

**subroutine: scan_for_mergers**

*Description:* Scan for and record mergers between nodes.
*Code lines:* 268
*Contained by:* module merger_tree_read
*Modules used:* cosmology_functions, dark_matter_halo_scales, galacticus_error, kepler_orbits

describe subroutine
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: 56

Contained by: module merger_tree_read

Modules used: node_subhalo_promotions

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.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: 167

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: 146

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: 65
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
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_state

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.
Code lines: 13
Contained by: module merger_trees_state_store
Modules used: galacticus_nodes

file: merger_trees.evolve.F90
Description: Contains a module which implements evolution of merger trees.
Code lines: 740

module: merger_trees_evolve
Description: Implements evolution of merger trees.
Code lines: 720
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: 205
18.1. Program units

**contained by:** module `merger_trees_evolve`
**modules used:**
- `cosmology_functions`
- `evolve_to_time_reports`
- `galacticus_display`
- `galacticus_error`
- `input_parameters`
- `merger_trees_evolve_node`
- `merger_trees_evolve_timesteps`
- `template`
- `string_handling`

**subroutine:** `merger_tree_evolve_to`
**description:** Evolves all properties of a merger tree to the specified time.
**code lines:** 307

**contained by:** module `merger_trees_evolve`
**modules used:**
- `galacticus_display`
- `galacticus_error`
- `input_parameters`
- `merger_trees_dump`
- `merger_trees_evolve_deadlock_status`
- `merger_trees_evolve_node`
- `merger_trees_evolve_timesteps`
- `node_component_hot_halo_standard`
- `string_handling`

**subroutine:** `perform_node_events`
**description:** Perform any events associated with `thisNode`.
**code lines:** 46

**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:**
- `node_branch_jump`
- `node_subhalo_promotion`
- `merger_tree_evolve_to`
- `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:** 797

**module:** `merger_trees_evolve_node`
**description:** Implements evolution of a single node in a merger tree.
**code lines:** 777

**contained by:** file `merger_trees.evolve.node.F90`
**modules used:**
- `fodeiv2`
- `galacticus_nodes`
- `iso_varying_string`
- `kind_numbers`

**used by:**
- `node_subhalo_promotion`
- `evolve_to_time`
- `merger_tree_evolve_to`
subroutine: events_node_merger
Description: Handles instances where thisNode is about to merge with its parent node.
Code lines: 102
Contained by: module merger_trees_evolve_node
Modules used: events_node_mergers_slh
galacticus_display
galacticus_error
module input_parameters
node_component_basic_standard
node_component_hot_halo_standard
node_component_hot_halo_very_simple
node_component_merging_statistics_recent
node_component_merging_statistics_standard
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: 98
Contained by: module merger_trees_evolve_node
Modules used: galactic_structure_radii
node_component_basic_non_evolving
node_component_basic_standard
node_component_black_hole_simple
node_component_black_hole_standard
node_component_dark_matter_profile_scale
node_component_dark_matter_profile_scale_shape
node_component_disk_exponential
node_component_disk_very_simple
node_component_formation_times_cole2000
node_component_hot_halo_standard
node_component_hot_halo_very_simple
node_component_inter_output_standard
node_component_satellite_standard
node_component_spheroid_standard
node_component_spin_preset

subroutine: tree_node_evolve
Description: Evolves thisNode to time endTime, or until evolution is interrupted.
Code lines: 205
Contained by: module merger_trees_evolve_node
Modules used: galacticus_calculations_resets
iso_c_binding
memory_management
node_component_basic_non_evolving
node_component_basic_standard
node_component_black_hole_simple
node_component_black_hole_standard
node_component_dark_matter_profile_scale
node_component_dark_matter_profile_scale_shape
node_component_disk_exponential
node_component_disk_very_simple
node_component_hot_halo_standard
node_component_hot_halo_very_simple
node_component_spheroid_standard
node_component_satellite_standard
node_component_satellite_very_simple
odeiv2_solver

subroutine: tree_node_evolve_error_analyzer
Description: Profiles ODE solver step sizes and errors.
18.1. Program units

subroutine: tree_node_evolve_initialize
Description: Initializes the tree evolving routines by reading in parameters
Code lines: 73
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: 112
Contained by: module merger_trees_evolve_node
Modules used: galacticus_display node_component_basic_non_evolving
node_component_basic_standard node_component_dark_matter_profile_scale
node_component_dark_matter_profile_scale_shape node_component_formation_times_cole2000
node_component_hot_halo_standard node_component_hot_halo_very_simple
node_component_merging_statistics_recent node_component_merging_statistics_standard
node_component_position_preset node_component_satellite_preset
node_component_spin_preset 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: 79

module: merger_tree_timesteps
Description: Implements calculations of timesteps for merger tree evolution.

Code lines: 59

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: 49

Contained by: module `merger_tree_timesteps`

Modules used:
- `iso_varying_string`
- `merger_tree_timesteps_record`
- `merger_tree_timesteps_satellite`
- `merger_tree_timesteps_simple`
- `merger_trees_evolve_timesteps_template`

**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: 379

**module: merger_tree_timesteps_history**

Description: Implements a simple time-stepping criterion for merger tree evolution.

Code lines: 358

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: 54

Contained by: module `merger_tree_timesteps_history`

Modules used:
- `galactic_structure_enclosed_masses`
- `galactic_structure_options`
- `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: 118

Contained by: module `merger_tree_timesteps_history`

Modules used:
- `cosmology_functions`
- `evolve_to_time_reports`
- `input_parameters`
- `iso_varying_string`
- `memory_management`
- `merger_trees_evolve_timesteps_template`
18.1. Program units

**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:* 268

**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:* 247

*Contained by:* file merger_trees.evolve.timesteps.record_evolution.F90

*Used by:* subroutine galacticus_merger_tree_output function time_step_get

**subroutine**: merger_tree_record_evolution_output

*Description:* Store Fourier-space halo profiles to the output file.

*Code lines:* 49

*Contained by:* module merger_tree_timesteps_record_evolution

*Modules used:*
- galacticus_hdf5
- galacticus_nodes
- galacticus_output_times
- iso_varying_string
- numerical_constants_astronomical
- string_handling

**subroutine**: merger_tree_record_evolution_store

*Description:* Store properties of the main progenitor galaxy.

*Code lines:* 30

*Contained by:* module merger_tree_timesteps_record_evolution

*Modules used:*
- galactic_structure_enclosed_masses
- galactic_structure_options
- galacticus_nodes
- 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:* 117

*Contained by:* module merger_tree_timesteps_record_evolution

*Modules used:*
- cosmology_functions
- evolve_to_time_reports
- input_parameters
- 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 time-stepping 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: 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: 227

module: merger_tree_timesteps_satellite
Code lines: 206
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_merging_process
Description: Process a satellite node which has undergone a merger with its host node.
Code lines: 76
Contained by: module merger_tree_timesteps_satellite
Modules used: galacticus_display galacticus_nodes iso_varying_string merger_trees_evolve_deadlock_status node_component_black_hole_simple node_component_black_hole_standard node_component_disk_exponential node_component_disk_very_simple 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 satellite_merging_remnant_sizes 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: 109

module: merger_tree_timesteps_simple
Description: Implements a simple time-stepping criterion for merger tree evolution.
Code lines: 89
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 timestepSimpleRelative$H^{-1}(t)$.
Code lines: 70
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 record_evolution satellite record_evolution satellite}

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: 180

module: merger_trees_millennium
Description: Handles reading of data from CSV files extracted from the Millennium Simulation database.
Code lines: 160
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.
18. Source Code Documentation

**Code lines:** 150  
**Contained by:** module `merger_trees_millennium`  
**Modules used:** `dates_and_times`, `galacticus_error`, `merger_tree_data_structure`

**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:** 173

**Module:** `merger_trees_simple`  
**Description:** Handles reading of data from CSV files of simple merger trees.  
**Code lines:** 153  
**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:** 143  
**Contained by:** module `merger_trees_simple`  
**Modules used:** `cosmological_parameters`, `file_utilities`, `iso_varying_string`, `merger_tree_data_structure`

**File:** `merger_trees.initialize.F90`  
**Description:** Contains a module which implements initialization of merger tree structures.  
**Code lines:** 95

**Module:** `merger_trees_initialize`  
**Description:** Implements initialization of merger tree structures.  
**Code lines:** 75  
**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:** 65  
**Contained by:** module `merger_trees_initialize`  
**Modules used:** `galacticus_nodes`, `node_component_basic_standard`, `node_component_dark_matter_profile_scale`, `node_component_dark_matter_profile_scale_shape`, `node_component formation_time cole2000`, `node_component_hot_halo scale_shape`, `node_component_hot_halo very simple recent`, `node_component_indices standard`, `node_component_merging_statistics standard`
18.1. Program units

node_component_satellite_standard  node_component_satellite_very_simple
node_component_spin_preset        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: 154

module: merger_tree_mass_accretion_history
Description: Outputs mass accretion histories of merger trees.
Code lines: 134
Contained by: file merger_trees.mass_accretion_history.F90
Modules used: io_hdf5
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: 100
Contained by: module merger_tree_mass_accretion_history
Modules used: galacticus_error  galacticus_hdf5
galacticus_nodes  input_parameters
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: 292

module: merger_tree_output_structure
18. Source Code Documentation

**Description:** Outputs the structure of entire merger trees.
**Code lines:** 272
**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:** 235
**Contained by:** module `merger_tree_output_structure`
**Modules used:** `dark_matter_halo_scales` `galacticus_nodes` `iso_varying_string` `node_component_dark_matter_profile_scale` `numerical_constants_astronomical` `galacticus_hdf5` `input_parameters` `memory_management` `node_component_dark_matter_profile_scale_shape` `string_handling`

**subroutine: merger_tree_structure_preclose**
**Description:** Close the merger tree structure group.
**Code lines:** 6
**Contained by:** module `merger_tree_output_structure`

**file: merger_trees.prune_branches.F90**
**Description:** Contains a module which prunes branches below a given mass threshold from merger trees.
**Code lines:** 138

**module: merger_trees_prune_branches**
**Description:** Prunes branches below a given mass threshold from merger trees.
**Code lines:** 118
**Contained by:** file `merger_trees.prune_branches.F90`
**Used by:** function `galacticus_task_evolve_tree`

**subroutine: merger_tree_prune_branches**
**Description:** Prune branches from `thisTree`.
**Code lines:** 96
**Contained by:** module `merger_trees_prune_branches`
**Modules used:** `galacticus_nodes` `input_parameters`

**file: merger_trees.prune_hierarchy.F90**
**Description:** Contains a module which prunes hierarchy below a given depth in merger trees.
**Code lines:** 122

**module: merger_trees_prune_hierarchy**
**Description:** Prunes hierarchy below a given depth in merger trees.
**Code lines:** 102
**Contained by:** file `merger_trees.prune_hierarchy.F90`
**Used by:** function `galacticus_task_evolve_tree`
18.1. Program units

**subroutine: merger_tree_prune_hierarchy**
*Description:* Prune hierarchy from thisTree.
*Code lines:* 83
*Contained by:* module merger_trees_prune_hierarchy
*Modules used:* galacticus_nodes

**file: merger_trees.regrid_times.F90**
*Description:* Contains a module which prunes branches below a given mass threshold from merger trees.
*Code lines:* 419

**module: merger_trees_regrid_times**
*Description:* Forces a merger tree onto a specified time grid.
*Code lines:* 399
*Contained by:* file merger_trees.regrid_times.F90
*Modules used:* iso_varying_string

**subroutine: merger_tree_regrid_time**
*Description:* Regrid times of halos in thisTree.
*Code lines:* 366
*Contained by:* module merger_trees_regrid_times
*Modules used:* cosmology_functions
gfs1
galacticus_nodes
kind_numbers
merger_trees_dump
numerical_ranges
critical_overdensity
galacticus_error
input_parameters
memory_management
numerical_interpolation

**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
*Contained by:* module merger_trees_render
*Modules used:* cosmology_functions
cosmology_functions
dark_matter_halo_scales
file_utilities
file_utilities
input_parameters
input_parameters
io_hdf5
io_hdf5
memory_management
memory_management
numerical_constants_astronomical

file: merger_trees.write.F90
Description: Contains a module which writes merger trees to file.
Code lines: 267

module: merger_trees_write
Description: Writes merger trees to file.
Code lines: 247
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: 218
Contained by: module merger_trees_write
Modules used: cosmological_parameters cosmology_functions
dates_and_times galacticus_nodes
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 expansion_factor_change subroutine cosmology_functions_matter_lambda_initialize
lambda_initialize
subroutine make_expansion_factor_table function dark_matter_halo_mass_accretion_time_zhao2009
linear_growth_factor_accretion_time_zhao2009
simple_tabulate
subroutine linear_growth_factor_accretion_time_zhao2009 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.
18.1. Program units

**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  
*Used by:* subroutine tree_node_evolve  

**module**: odeiv2  
*Description:* Contains an interface to the GNU Scientific Library ODEIV2 differential equation solvers.  
*Code lines:* 495  
*Contained by:* file numerical.ODE_solver.ODEIV2.wrapper.F90  
*Modules used:* fodeiv2  

**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
18. Source Code Documentation

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
Contained by: module fodeiv2

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
18. Source Code Documentation

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

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
18.1. Program units

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 tree_node_odes module ode_solver
module odeiv2_solver

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
subroutine cosmology_functions_matter_lambda_initialize subroutine galacticus_output_time_index
function hot_halo_profile_radial_moment_cored_isothermal_get subroutine merger_tree_read_initialize
subroutine read_node_data subroutine tree_node_is_accurate
function search_array_for_closest subroutine satellite_merging_remnant_size_cole2000
subroutine satellite_merging_remnant_size_covington2008 subroutine critical_overdensity_kitayama_suto1996_initialize
function g_2_integrated subroutine power_spectrum_nonlinear_cosmicemu
subroutine transfer_function_file_read subroutine virial_density_bryan_norman_initialize
subroutine virial_density_kitayama_suto1996_initialize subroutine test_comparison
subroutine assert_double_1d_array subroutine assert_double_2d_array
subroutine assert_double_3d_array subroutine assert_double_4d_array
subroutine assert_double_5d_array subroutine assert_double_scalar
subroutine assert_real_1d_array subroutine assert_real_scalar

interface: values_agree
Code lines: 3
Contained by: module numerical_comparison

function: values_agree_double
18. Source Code Documentation

*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.

*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:* 65

**module: numerical_constants_astronomical**

*Description:* Contains various useful astronomical constants.

*Code lines:* 45

*Contained by:* file `numerical.constants.astronomical.F90`

*Modules used:* `numerical_constants_atomic` `numerical_constants_units`

*Used by:* function `bondi_hoyle_lyttleton_accretion_rate`

function `accretion_disk_jet_power_shakura_sunyaev`

function `black_hole_eddington_accretion_rate`

subroutine `chemical_densities_cie_file_interpolate`

function `electron_density_cie_file_logtemperature_interpolate`

function `cooling_function_cie_file_logtemperature_interpolate`

function `black_hole_binary_separation_growth_rate_standard`

function `chemicals_mass_to_density_conversion`

function `electron_density_cie_file_interpolate`

function `cooling_function_cie_file_interpolate`

subroutine `cooling_radius_hot_halo_output_names`
18.1. Program units

subroutine cooling_rate_hot_halo_output_names
function comoving_volume_element_time
function h_0_invgyr
function dark_matter_halo_dynamical_timescale
function dark_matter_profile_freefall_radius_einasto
function dark_matter_profile_freefall_radius_increase_rate_nfw
function dark_matter_profile_freefall_radius_increase_rate_isothermal
function bar_instability_timescale_eln

subroutine meta_tree_timing_output

subroutine make_output_group

subroutine galacticus_merger_tree_output_filter_lightcone_initialize
subroutine galacticus_output_tree_density_contrast_names
subroutine galacticus_output_tree_lightcone_names
subroutine galacticus_linear_power_spectrum_output
subroutine galacticus_output_tree_rotation_curve_names
subroutine galacticus_output_tree_velocity_dispersion_names
subroutine halo_mass_function_output

subroutine igm_state_electron_scattering_integrand
subroutine merger_tree_read_initialize
subroutine merger_tree_record_evolution_output
subroutine merger_trees_simple_process

subroutine merger_tree_structure_output
subroutine merger_tree_write
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

function cooling_time_simple
function comoving_distance_integrand
function omega_radiation
function dark_matter_halo_virial_temperature
function dark_matter_profile_freefall_radius_increase_rate_einasto
function dark_matter_profile_freefall_radius_nfw
function dark_matter_profile_freefall_radius_isothermal

subroutine galacticus_meta_evolver_profiler_output
subroutine galacticus_merger_tree_output_make_group
subroutine galacticus_output_tree_density_contrast_names
subroutine galacticus_output_tree_lightcone_names
subroutine galacticus_linear_power_spectrum_output
subroutine galacticus_output_tree_rotation_curve_names
subroutine galacticus_output_tree_velocity_dispersion_names
subroutine halo_mass_function_output

subroutine intergalactic_medium_state_recfast_initialize
subroutine merger_tree_history_write
subroutine merger_trees_millennium_process
subroutine merger_trees_mass_accretion_history_output
subroutine merger_trees_render_dump
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_standard_formation
subroutine node_component_hot_halo_standard_rate_compute
subroutine node_component_spheroid_standard_rate_compute
function ram_pressure_stripping_mass_loss_rate_disk_simple
function imf_energy_input_rate_noninstantaneous
function imf_metal_yield_rate_noninstantaneous
function imf_recycling_rate_noninstantaneous
subroutine star_formation_rate_surface_density_disks_br_initialize
function star_formation_timescale_disk_dynamical_time
function star_formation_timescale_spheroid_dynamical_time
subroutine supernovae_population_iii_hegerwoosley_initialize
function filter_luminosity_integrand_ab
subroutine stellar_population_spectra_file_tabulation
function ideal_gas_sound_speed

file: numerical.constants.atomic.F90
Description: Contains a module of useful atomic constants.
Code lines: 39

module: numerical.constants.atomic
Description: Contains various useful atomic constants.
Code lines: 19
Contained by: file numerical.constants.atomic.F90
Modules used: fgsl
Used by: subroutine stellar_population_spectra_postprocess_lyc_suppress
subroutine stellar_population_spectra_postprocess_madau1995
subroutine stellar_population_spectra_postprocess_meiksin2006

file: numerical.constants.math.F90
Description: Contains a module of useful mathematical constants.
Code lines: 40

module: numerical.constants.math
Description: Contains various useful mathematical constants.
Code lines: 20
Contained by: file numerical.constants.math.F90
Modules used: fgsl
Used by: program tests_excursion_sets
function black_hole_binary_recoil_standard
function black_hole_static_radius_spin
function cooling_rate_cole2000
function cooling_rate_white_frenk
function critical_density
function dark_matter_halo_virial_radius
function density_einasto_scale_free
function fourier_profile_integrand_einasto
function nfw_profile_energy
function cmin
function dark_matter_profile_rotation_curve_gradient_task
function mean_density_contrast_root
function hot_halo_profile_rotation_curve_gradient_task
function hot_halo_profile_radial_moment_cored_isothermal_get
module generalized_press_schechter_branching
module numerical_constants_units
function node_component_disk_exponential_rotation_curve_bessel_factors
subroutine power_spectrum_compute
function star_formation_rate_surface_density_disk_br
function star_formation_rate_surface_density_disk_ks
function star_formation_timescale_disk_integrated_sd
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 dark_matter_profile_rotation_normalization_einasto
subroutine energy_table_make
function density_nfw_scale_free
function dark_matter_profile_density_isothermal
function inversesigman
function galactic_structure_radius_initial_derivative_adiabatic_solver
function spherical_shell_solid_angle_in_cylcinder
function hot_halo_density_cored_isothermal_get
function gaussian_distribution
module modified_press_schechter_branching
function mass_distribution_mass_enc_by_sphere_spherical
function node_component_disk_exponential_rttn_crv_grdnt_bssl_fctrs
function star_formation_feedback_disk_outflow_rate_creasey2012
function star_formation_rate_surface_density_disk_ks
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 subroutine transfer_function_null_make
subroutine virial_density_bryan_norman subroutine virial_density_kitayama-suto1996
program tests_cosmic_age_dark_energy_closed program test_coordinate_systems
program test_integration program tests_power_spectrum
program tests_sigma program tests_sphericalCollapse_density
program tests_sphericalCollapse_density_omega_012 program tests_sphericalCollapse_density_omega_023
program tests_sphericalCollapse_density_lambda program tests_sphericalCollapse_density_open
program tests_sphericalCollapse_flat

file: numerical.constants.physical.F90
Description: Contains a module of useful physical constants.
Code lines: 54

module: numerical_constants_physical
Description: Contains various useful physical constants.
Code lines: 34
Contained by: file numerical.constants.physical.F90
Modules used: fgs1
Used by: function bondi_hoyle_lyttleton_accretion_radius
function atomic_cross_section_compton
function atomic_rate_ionization_collisional_verner
function black_hole_binary_separation_growth_rate_standard
function black_hole_gravitational_radius
function black_hole_isco_specific_energy_node
subroutine chemical_hydrogen_rate_h2_gamma_to_2h
subroutine chemical_hydrogen_rate_h2_gamma_to_3h
subroutine chemical_hydrogen_rate_h2_plus_gamma_to_2hplus_electron
subroutine chemical_hydrogen_rate_h2_minus_gamma_to_h_minus_2h
subroutine chemical_hydrogen_rate_h2_plus_minus_h_to_2h
subroutine chemical_hydrogen_rate_h2_plus_plus_gamma_to_2hplus_electron
subroutine chemical_hydrogen_rate_h_plus_plus_electron_to_2hplus_electron
subroutine chemical_hydrogen_rate_hminus_gamma_to_h_electron

numerical_constants_prefixes
function bondi_hoyle_lyttleton_accretion_rate
function accretion_disk_jet_power_eddington
function atomic_cross_section_ionization_photo_verner
function black_hole_binary_initial_radius_volonteri_2003
function black_hole_eddington_accretion_rate
function black_hole_isco_specific_angular_momentum
subroutine chemical_hydrogen_rate_h2__gamma_to_2h
subroutine chemical_hydrogen_rate_h2__gamma_to_2hstar_to_2h
subroutine chemical_hydrogen_rate_h2__hplus_to_h2plus_h
subroutine chemical_hydrogen_rate_h2_plus_plus_h2_plus_plus_electron
subroutine chemical_hydrogen_rate_h_plus_plus_electron_to_2hplus_electron
subroutine chemical_hydrogen_rate_hplus_to_2hplus_photon
subroutine chemical_hydrogen_rate_hminus_to_2h_electron
18.1. Program units

subroutine chemical_hydrogen_rate_hminus_hplus_to_h2plus_electron
function cross_section_h2_gamma_to_h2plus_electron
function cross_section_h2plus_gamma_to_h_hplus
function h_hminus_to_h2_electron_rate_coefficient
subroutine cooling_function_cmb_compton
function cooling_specific_angular_momentum_constant_rotation
function comoving_volume_element_redshift
function critical_density
function dark_matter_halo_virial_temperature
function dark_matter_halo-angular_momentum
function dark_matter_profile_freefall_radius_einasto
function dark_matter_profile_potential_einasto
function dark_matter_profile_circular_velocity_nfw
function dark_matter_profile_rotation_curve_task
subroutine solve_for_radius
function velocity DISPERSION_integrand
function hot_halo_profile_rotation_curve_task
function igm_state_electron_scattering_integrand
module chemical_structures
subroutine kepler_orbits_propagate
subroutine node_component_black_hole_simple_rate_compute
function node_component_black_hole_simple_rotation_curve
subroutine node_component_black_hole_standard_rate_compute
function node_component_black_hole_standard_potential
function node_component_black_hole_standard_rotation_curve_gradient
function cross_section_h2_gamma_to_2h
function cross_section_h2plus_gamma_to_2hplus_electron
function cross_section_hminus_gamma_to_h_electron
function hminus_electron_to_h_2electron_rate_coefficient
function cooling_time_simple
function comoving_volume_element_rotation
function omega_radiation
function dark_matter_halo_virial_velocity
function dark_matter_profile_circular_velocity_einasto
function dark_matter_profile_freefall_radius_einasto
function radius_from_specific_angular_momentum_einasto
function dark_matter_profile_rotation_curve_gradient_task
function bar_instability_timescale_eln
subroutine galactic_structure_radii_initial_adiabatic_compute_factors
function hot_halo_profile_rotation_curve_gradient_task
function hot_halo_ram_pressure_stripping_radius_solver
subroutine assign_spin_parameters
function kepler_orbits_energy
function kepler_orbits_velocity_scale
function node_component_black_hole_simple_potential
function node_component_black_hole_simple_rotation_curve_gradient
subroutine node_component_black_hole_standard_triple_interaction
function node_component_black_hole_standard_rotation_curve
subroutine node_component_disk_exponential_radius_solver
function radiation_integrate_over_cross_section  
function ram_pressure_stripping_mass_loss_rate_disk_simple  
subroutine satellite_merging_remnant_size_cole2000  
subroutine satellite_merging_remnant_size_coffington2008  
subroutine satellite_merging_remnant_progenitor_properties_cole2000  
subroutine satellite_merging_remnant_progenitor_properties_standard  
function satellite_orbit_convert_to_current_potential  
function star_formation_rate_surface_density_disk_br  
subroutine star_formation_rate_surface_density_disk_br_initialize  
function star_formation_rate_surface_density_disk_ks  
subroutine star_formation_rate_surface_density_disk_ks_initialize  
program tests_kepler_orbits  
function ideal_gas_sound_speed  
function blackbody_emission

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_iswilm2000
        subroutine cooling_function_temperature_slope_molecular_hydrogen_gpe
        module numerical_constants_physical
        subroutine store_unit_attributes_ratre
        function star_formation_rate_surface_density_disk_kmt09
        subroutine star_formation_rate_surface_density_disks_exschmidt_initialize
        function atomic_cross_section_compton
        subroutine number_density_critical_over_number_density_hydrogen
        subroutine number_density_critical_over_number_density_hydrogen
        module numerical_constants_units
        function star_formation_feedback_disk_outflow_rate_creasey2012_integrand
        subroutine star_formation_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
Used by: program xray_absorption_iswilm2000
        function atomic_cross_section_ionization_photo_verner
        subroutine chemical_hydrogen_rate_h2_gamma_to_2h
        subroutine chemical_hydrogen_rate_h2_gamma_to_h2plus_electron
        module numerical_constants_prefixes
        function atomic_cross_section_compton
        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_-gamma_to_h2star_to_2h
subroutine chemical_hydrogen_rate_h2_-hplus_to_h2plus_h
subroutine chemical_hydrogen_rate_-h2plus_gamma_to_h_hplus
subroutine chemical_hydrogen_rate_h_-hplus_to_h2plus_photon
subroutine chemical_hydrogen_rate_-hminus_h_to_2h_electron
function cross_section_h2_gamma_to_2h
function cross_section_h2plus_gamma_to_2h
function cross_section_h2plus_gamma_to_h_hplus
function h_hminus_to_2h_electron_rate_coefficient
module numerical_constants_-astronomical
subroutine radiation_flux_cmb

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

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: fgsl
Used by: program optimal_sampling_smf subroutine make_distance_table
subroutine energy_table_make subroutine fourier_profile_table_make

18.1. Program units
function \texttt{freefall\_time\_scale\_free} function \texttt{freefall\_time\_scale\_free} function \texttt{galactic\_structure\_velocity\_dispersion}

subroutine \texttt{galacticus\_output\_tree\_velocity\_dispersion} subroutine \texttt{galacticus\_output\_trees\_line\_of\_sight\_velocity\_dispersion} function \texttt{generalized\_press\_schechter\_-branch\_mass\_root}

subroutine \texttt{igm\_state\_electron\_scattering\_tabulate} subroutine \texttt{merger\_tree\_build\_initialize} subroutine \texttt{igm\_state\_electron\_scattering\_tabulate} subroutine \texttt{merger\_tree\_build\_initialize} subroutine \texttt{igm\_state\_electron\_scattering\_tabulate} subroutine \texttt{merger\_tree\_build\_initialize}

function \texttt{generalized\_press\_schechter\_-branching\_probability} function \texttt{modified\_press\_schechter\_-branching\_probability} function \texttt{generalized\_press\_schechter\_-branching\_probability} function \texttt{modified\_press\_schechter\_-branching\_probability}

function \texttt{modified\_press\_schechter\_-branch\_mass\_root} function \texttt{merger\_tree\_construct\_mass\_-function\_sampling\_stellar\_mf} function \texttt{modified\_press\_schechter\_-branch\_mass\_root} function \texttt{merger\_tree\_construct\_mass\_-function\_sampling\_stellar\_mf}

function \texttt{mass\_distribution\_mass\_enc\_-by\_sphere\_spherical} function \texttt{radiation\_integrate\_over\_-cross\_section} function \texttt{mass\_distribution\_mass\_enc\_-by\_sphere\_spherical} function \texttt{radiation\_integrate\_over\_-cross\_section}

function \texttt{imf\_energy\_input\_rate\_-noninstantaneous} function \texttt{imf\_metal\_yield\_rate\_-noninstantaneous} function \texttt{imf\_energy\_input\_rate\_-noninstantaneous} function \texttt{imf\_metal\_yield\_rate\_-noninstantaneous}

function \texttt{imf\_recycling\_rate\_-noninstantaneous} function \texttt{star\_formation\_feedback\_disk\_-outflow\_rate\_creasey2012} function \texttt{imf\_recycling\_rate\_-noninstantaneous} function \texttt{star\_formation\_feedback\_disk\_-outflow\_rate\_creasey2012}

function \texttt{star\_formation\_timescale\_-disk\_integrated\_sd} function \texttt{stellar\_feedback\_cumulative\_-energy\_input\_standard} function \texttt{star\_formation\_timescale\_-disk\_integrated\_sd} function \texttt{stellar\_feedback\_cumulative\_-energy\_input\_standard}

function \texttt{stellar\_population\_luminosity} function \texttt{variance\_integral} function \texttt{stellar\_population\_luminosity} function \texttt{variance\_integral}

function \texttt{g\_2\_integrated} function \texttt{halo\_mass\_fraction\_integrated} function \texttt{g\_2\_integrated} function \texttt{halo\_mass\_fraction\_integrated}

function \texttt{halo\_mass\_function\_integrated} function \texttt{t\_collapse} function \texttt{halo\_mass\_function\_integrated} function \texttt{t\_collapse}

program \texttt{test\_integration} program \texttt{test\_integration} program \texttt{test\_integration}

\textbf{function: integrate}\n\textit{Description:} Integrates the supplied \texttt{integrand} function.
\textit{Code lines:} 84
\textit{Contained by:} module \texttt{numerical\_integration}
\textit{Modules used:} \texttt{galacticus\_error} \texttt{iso\_c\_binding}

\textbf{subroutine: integrate\_done}\n\textit{Description:} Frees up integration objects that are no longer required.
\textit{Code lines:} 9
\textit{Contained by:} module \texttt{numerical\_integration}

\textbf{subroutine: integration\_gsl\_error\_handler}\n\textit{Description:} Handle errors from the GSL library during integration.
\textit{Code lines:} 8
\textit{Contained by:} module \texttt{numerical\_integration}
\textit{Modules used:} \texttt{iso\_c\_binding}

\textbf{file: numerical\_interpolation\_.2D.irregular.F90}\n\textit{Description:} Contains a module which implements a convenient interface to the BIVAR 2D interpolation on irregularly spaced points package.
\textit{Code lines:} 130

\textbf{module: numerical\_interpolation\_2d\_irregular}
18.1. Program units

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: 319

module: numerical_interpolation

Description: A simple interface to the GNU Scientific Library interpolation routines.

Code lines: 298

Contained by: file numerical.interpolation.F90

Modules used: fgsl  
iso_c_binding

Used by: subroutine get_interpolation  
subroutine get_interpolation

function cosmology_age_matter_dark_-energy  
function expansion_factor_matter_dark_-energy

subroutine make_expansion_factor_table  
function comoving_distance_conversion_-matter_lambda

function comoving_distance_matter_-lambda  
function cosmology_age_matter_lambda

function expansion_factor_matter_lambda  
subroutine make_distance_table

subroutine make_expansion_factor_table  
function time_from_comoving_distance_-matter_lambda

subroutine dark_matter_profile_-einstein  
function dark_matter_profile_energy_-einstein
function dark_matter_profile_energy_growth_rate_einasto
function dark_matter_profile_freefall_radius_increase_rate_einasto
subroutine energy_table_make
function radius_from_specific-angular_momentum_scale_free
function filter_response

function intergalactic_medium_temperature_file
subroutine merger_tree_history_store
subroutine merger_tree_record_evolution_store
subroutine merger_tree_regrid_time
subroutine history_increment
function table_generic_1d_interpolate

subroutine radiation_igb_file_flux
function imf_metal_yield_rate_noninstantaneous
function snepopiii_cumulative_energy_hegerwoosley
function stellar_population_luminosity

function stellar_population_spectra_file_interpolate
function critical_overdensity_mass_scaling_wdm
function excursion_sets_first_crossing_probability_farahi
subroutine excursion_sets_first_crossing_rate_tabulate_farahi
function excursion_sets_first_crossing_probability_zhang_hui
function halo_mass_function_differential_tinker2008
function linear_growth_factor

subroutine linear_growth_factor_simple_tabulate
function primordial_power_spectrum

function dark_matter_profile_freefall_radius_einasto
function dark_matter_profile_kspace_einasto
subroutine fourier_profile_table_make
subroutine radius_from_specific-angular_momentum_table_make
function intergalactic_medium_temperature_file
subroutine merger_tree_build_initialize

subroutine merger_tree_timestep_history
subroutine merger_tree_timestep_record_evolution
subroutine merger_tree_write
subroutine stellar_tracks_interpolation_get
subroutine stellar_population_properties_rates_noninstantaneous

function critical_overdensity_mass_scaling_gradient_wdm
subroutine excursion_sets_first_crossing_farahi_read_file
function excursion_sets_first_crossing_farahi
function excursion_sets_non_crossing_rate_farahi
function excursion_sets_first_crossing_probability_zhang_hui_high
subroutine interpolate_in_wavenumber

function linear_growth_factor_logarithmic_derivative
function power_spectrum_nonlinear_cosmicemu
function primordial_power_spectrum_logarithmic_derivative
function transfer_function

function transfer_function_-logarithmic_derivative

program test_interpolation

function: interpolate
Description: Perform an interpolation of x into xArray() and return the corresponding value in yArray().
Code lines: 93
Contained by: module numerical_interpolation
Modules used: galacticus_error iso_varying_string

function: interpolate_derivative
Description: Perform an interpolation of x into xArray() and return the corresponding first derivative of yArray().
Code lines: 70
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 iInterpolate and interpolating factors interpolationFactors for array yArray, return a linearly interpolated value.
Code lines: 10
Contained by: module numerical_interpolation

function: interpolate_linear_generate_factors
Description: Return interpolating factors for linear interpolation in the array xArray() given the index in the array which brackets value x.
Code lines: 12
Contained by: module numerical_interpolation

function: interpolate_linear_generate_gradient_factors
Description: Return interpolating factors for linear interpolation in the array xArray() given the index in the array which brackets value x.
Code lines: 12
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: 51
Contained by: module numerical_interpolation
Modules used: galacticus_error

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: program test_meshes

subroutine: meshes_apply_point
Description: Apply a point to a mesh.
Code lines: 60
Contained by: module meshes
Modules used: galacticus_error iso_c_binding

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: 114

module: pseudo_random
Description: Implements pseudo-random numbers.
Code lines: 94
Contained by: file numerical.random.F90
Modules used: fgsl
Used by: subroutine black_hole_binary_recoil_velocity_standard
function halo_spin_distribution_bett2007
subroutine halo_spin_distribution_lognormal_state_store
subroutine halo_spin_distribution_lognormal_state_retrieve
subroutine merger_tree_build_cole2000_state_retrieve
subroutine merger_tree_build_do_cole2000
subroutine satellite_time_until_merging_lacey_cole_tormen_state_retrieve
function virial_orbital_parameters_wetzel2010
subroutine black_hole_binary_recoil_velocity_standard_state_retrieve
subroutine black_hole_binary_recoil_velocity_standard_state_store
function halo_spin_distribution_bett2007_state_retrieve
subroutine halo_spin_distribution_bett2007_state_store
subroutine halo_spin_distribution_lognormal_state_store
subroutine merger_tree_build_cole2000_state_store
subroutine satellite_time_until_merging_lacey_cole_tormen_state_store
function virial_orbital_parameters_benson2005
subroutine virial_orbital_parameters_benson2005_state_retrieve
subroutine virial_orbital_parameters_benson2005_state_store
subroutine virial_orbital_parameters_wetzel2010
subroutine virial_orbital_parameters_wetzel2010_state_retrieve
subroutine virial_orbital_parameters_wetzel2010_state_store

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: 47
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 satellite_time_until_merging_lacey_cole_tormen

subroutine: gaussian_random_free
Description: Frees a pseudo-random sequence object.
Code lines: 7
Contained by: module gaussian_random

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.
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 tests_excursion_sets
program xray_absorption_ism_wilms2000
subroutine make_distance_table
subroutine dark_matter_profile_<
einasto_freefall_tabulate
subroutine fourier_profile_table_make
subroutine galacticus_meta_evolver_-profile
subroutine star_formation_histories_-metallicity_split_initialize
program optimal_sampling_smf
subroutine make_expansion_factor_table
subroutine make_expansion_factor_table
subroutine energy_table_make
subroutine radius_from_specific_-
angular_momentum_table_make
subroutine galacticus_linear_power_-spectrum_output
subroutine star_formation_history_-metallicity_split_make_history
18.1. Program units

subroutine halo_mass_function_compute subroutine merger_tree_build_initialize
subroutine merger_tree_timestep_history subroutine merger_tree_timestep_record_evolution
subroutine merger_tree_regrid_time subroutine history_combine
subroutine history_create subroutine history_extend
subroutine history_timesteps subroutine table_linear_1d_create
subroutine table_linear_cspline_1d_create subroutine power_spectrum_compute
function imf_energy_input_rate_noninstantaneous subroutine excursion_sets_first_crossing_probability_farah
function imf_reycling_rate_noninstantaneous subroutine excursion_sets_first_crossing_rate_tabulate_farahi
function excursion_sets_first_crossing_probability_zhang_hui subroutine power_spectrum_power_law_tabulate
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 subroutine tests_sigma
program test_make_ranges program tests_sigma

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: 351

module: root_finder
Description: Implements root finding.
Code lines: 331
Contained by: file numerical.root_finder.F90
Modules used: f gsl
Used by: function black_hole_binary_initial_radius_tidal_radius
function epoch_of_matter_dark_energy_enclosing_mass
function epoch_of_matter_domination_matter_dark_energy
function dark_matter_profile_concentration_nfw1996
function imf_energy_input_rate_noninstantaneous
function imf_reycling_rate_noninstantaneous
function excursion_sets_first_crossing_probability_farah
function excursion_sets_first_crossing_probability_zhang_hui
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
program tests_sigma

function cooling_radius_simple
function epoch_of_matter_domination_matter_dark_energy
function galactic_structure_radius_enclosing_mass
18. Source Code Documentation

- function `galactic_structure_radius_initial_adiabatic`
- subroutine `galacticus_output_tree-density_contrast`
- function `generalized_press_schechter_branch_mass`
- subroutine `assign_scale_radii`
- function `virial_orbital_parameters_wetzel2010`
- subroutine `satellite_orbit_pericenter-phase_space_coordinates`
- subroutine `make_table`
- function `perturbation_maximum_radius`

function: `root_finder_find`
Description: Finds the root of the supplied root function.
Code lines: 130
Contained by: module `root_finder`
Modules used: `galacticus_error` iso_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: 33
Contained by: module `root_finder`

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
18.1. Program units

<table>
<thead>
<tr>
<th>Contained by:</th>
<th>module root_finder</th>
</tr>
</thead>
</table>

**type: rootfinder**

*Description:* Type containing all objects required when calling the FGSL root solver function.

*Code lines:* 66

*Contained by:*

<table>
<thead>
<tr>
<th>module root_finder</th>
</tr>
</thead>
</table>

**file: numerical.search.F90**

*Description:* Contains a module which implements searching of ordered arrays.

*Code lines:* 172

**module: arrays_search**

*Description:* Implements searching of ordered arrays.

*Code lines:* 151

*Contained by:*

<table>
<thead>
<tr>
<th>file numerical.search.F90</th>
</tr>
</thead>
</table>

*Used by:*

<table>
<thead>
<tr>
<th>subroutine galacticus_meta_evolver_profile</th>
<th>subroutine galacticus_merger_tree_profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>subroutine starFormation_history_record_metallcity_split</td>
<td>subroutine output_filter_lightcone</td>
</tr>
<tr>
<td>function galacticus_output_time_index</td>
<td>function galacticus_next_output_time</td>
</tr>
<tr>
<td>function galacticus_previous_output_time</td>
<td>function descendant_node_sort_index</td>
</tr>
<tr>
<td>subroutine read_node_data</td>
<td>subroutine history_combine</td>
</tr>
<tr>
<td>program test_search</td>
<td>subroutine delete_integer_scalar_vs</td>
</tr>
<tr>
<td>subroutine set_integer_scalar_vs</td>
<td>function value_integer_scalar_vs</td>
</tr>
</tbody>
</table>

**interface: search_array**

*Description:* Generic interface for array searching routines.

*Code lines:* 5

*Contained by:*

<table>
<thead>
<tr>
<th>module arrays_search</th>
</tr>
</thead>
</table>

**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:* 10

*Contained by:*

| module arrays_search |

**Modules used:**

<table>
<thead>
<tr>
<th>subroutine fgsl</th>
</tr>
</thead>
</table>

**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:* 30

*Contained by:*

| module arrays_search |

*Modules used:*

<table>
<thead>
<tr>
<th>subroutine fgsl</th>
</tr>
</thead>
<tbody>
<tr>
<td>galacticus_error</td>
</tr>
<tr>
<td>numerical_comparison</td>
</tr>
</tbody>
</table>

**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:** 43  
**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:** 45  
**Contained by:** module `arrays_search`  
**Modules used:** `iso_varying_string`

**file: numerical.sort.F90**

**Description:** Contains a module which implements sorting sequences.

**Code lines:** 152

**module: sort**

**Description:** Implements sorting.

**Code lines:** 132  
**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 `merger_tree_write` program `test_sort`

**function: compare_double**

**Description:** Comparison function for double precision data.

**Code lines:** 10  
**Contained by:** module `sort`

**function: compare_integer**

**Description:** Comparison function for integer data.

**Code lines:** 10  
**Contained by:** module `sort`

**function: compare_integer8**

**Description:** Comparison function for integer data.

**Code lines:** 10  
**Contained by:** module `sort`

**interface: sort_do**

**Description:** Generic interface to in-place sort routines.

**Code lines:** 4  
**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_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: 3
Contained by: module sort

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: 978

module: abundances_structure
Description: Defines the abundances structure used for describing elemental abundances in GALACTICUS.
Code lines: 958
Contained by: file objects.abundances.F90
Modules used: iso_varying_string numerical_constants_astronomical
Used by: module accretion_halos module accretion_halos_null
module accretion_halos_simple subroutine chemical_densities_cie_file
subroutine chemical_densities_cie_file_file_interpolate
function electron_density_cie_file_function electron_density_cie_file_interpolate
function electron_density_cie_file_interpolate logtemperature_interpolate
function electron_density_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_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_density_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 galacticus_nodes

subroutine node_component_disk_exponential_initialize
subroutine node_component_disk_exponential_rate_compute
subroutine node_component_disk_exponential_scale_set
subroutine node_component_hot_halo_standard_initialize

function electron_density_temperature_log_slope_cie_file
subroutine chemical_densities_atomic_cie_cloudy

function electron_density_atomic_cie_cloudy
function electron_density_temperature_log_slope_atomic_cie_cloudy
function cooling_function_cie_file_interpolate
subroutine cooling_function_density_slope_cie_file
subroutine cooling_function_cmb_compton

subroutine cooling_function_temperature_slope_cmb_compton
subroutine cooling_function_atomic_cie_cloudy
subroutine cooling_function_density_slope_molecular_hydrogen_gp
subroutine cooling_function_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 galacticus_nodes

subroutine node_component_disk_exponential_initialize
subroutine node_component_disk_exponential_rate_compute
subroutine node_component_disk_exponential_scale_set
subroutine node_component_hot_halo_standard_initialize
18.1. Program units

subroutine node_component_hot_halo_standard_outflowing_abundances_rate
subroutine node_component_hot_halo_standard_push_to_cooling_pipes
subroutine node_component_hot_halo_standard_rate_compute
subroutine node_component_hot_halo_standard_scale_set
subroutine node_component_hot_halo_standard_tree_initialize
subroutine node_component_spheroid_standard_energy_gas_input_rate
subroutine node_component_spheroid_standard_mass_gas_sink_rate
subroutine node_component_spheroid_standard_scale_set
subroutine node_component_spheroid_standard_satellite_merger
module star_formation_imf
function imf_select_disk_spheroid
function starFormation_rate_surface_density_disk_br
function starFormation_rate_surface_density_disk_kmt09
function starFormation_rate_surface_density_disk_ks
module stellar_population_luminosities
subroutine stellar_population_properties
function stellar_population_luminosities_get
function stellar_population_spectra_conroy_get
function stellar_population_spectra_file_get
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
18. Source Code Documentation

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: memory_management

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
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
18.1. Program units

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: 49
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
18. Source Code Documentation

**Description:** Return an element-by-element \texttt{max()} on two abundances objects.

**Code lines:** 9

**Contained by:** module \texttt{abundances\_structure}

**Function:** \texttt{abundances\_multiply}

**Description:** Multiply an abundances object by a scalar.

**Code lines:** 12

**Contained by:** module \texttt{abundances\_structure}

**Function:** \texttt{abundances\_multiply\_switched}

**Description:** Multiply a scalar by an abundances object.

**Code lines:** 9

**Contained by:** module \texttt{abundances\_structure}

**Function:** \texttt{abundances\_names}

**Description:** Return a name for the specified entry in the abundances structure.

**Code lines:** 23

**Contained by:** module \texttt{abundances\_structure}

**Modules used:** \texttt{galacticus\_error}

**Subroutine:** \texttt{abundances\_output}

**Description:** Store an abundances object in the output buffers.

**Code lines:** 16

**Contained by:** module \texttt{abundances\_structure}

**Subroutine:** \texttt{abundances\_output\_count}

**Description:** Increment the output count to account for an abundances object.

**Code lines:** 9

**Contained by:** module \texttt{abundances\_structure}

**Subroutine:** \texttt{abundances\_output\_names}

**Description:** Assign names to output buffers for an abundances object.

**Code lines:** 25

**Contained by:** module \texttt{abundances\_structure}

**Function:** \texttt{abundances\_property\_count}

**Description:** Return the number of properties required to track abundances. This is equal to the number of elements tracked, \texttt{elementsCount}, plus one since we always track a total metallicity.

**Code lines:** 10

**Contained by:** module \texttt{abundances\_structure}

**Subroutine:** \texttt{abundances\_read\_raw}

**Description:** Read an abundances object from binary.

**Code lines:** 12

**Contained by:** module \texttt{abundances\_structure}

**Subroutine:** \texttt{abundances\_reset}

**Description:** Reset an abundances object.

**Code lines:** 11
18.1. Program units

**subroutine**: abundances_serialize

*Description:* Pack abundances from an array into an abundances structure.
*Code lines:* 18
*Contained by:* module abundances_structure

**subroutine**: abundances_set_metallicity

*Description:* Set the metallicity of the self structure to metallicity.
*Code lines:* 73
*Contained by:* module abundances_structure
*Modules used:* atomic_data galacticus_error

**subroutine**: abundances_set_to_unity

*Description:* Set an abundances object to unity.
*Code lines:* 13
*Contained by:* module abundances_structure

**function**: 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:* 628

**module**: chemical_abundances_structure

*Description:* Defines the structure used for describing chemical abundances in GALACTICUS.
*Code lines:* 608
*Contained by:* file objects.chemical_abundances.F90
*Modules used:* iso_varying_string
*Used by:* module accretion_halos

subroutine halo_baryonic_accreted_-chemicals_null_get

subroutine halo_baryonic_accretion_rate_chemicals_null_get

subroutine get_chemical_masses

subroutine halo_baryonic_accretion_rate_chemicals_simple_get

subroutine accretion_halos_simple_initialize

subroutine halo_baryonic_accreted_-chemicals_simple_get

module chemical_reaction_rates
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.
18.1. Program units

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: 56
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: 152
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

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

892
18.1. Program units

```
18.1. Program units

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
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.
18.1. Program units

**module:** coordinates

*Description:* Implements the coordinates class.

*Code lines:* 505

*Contained by:* file objects.coordinates.F90

*Used by:* function mass_distribution_density_null  
  function mass_distribution_mass_enc_by_sphere_spherical_integrand  
  function mass_distribution_potential_null  
  program test_mass_distributions

**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:** coordinatenequidistance

*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

*Description:* Assign a 3-component vector to a coordinate object.

*Code lines:* 8

*Contained by:* module coordinates

**subroutine:** coordinates_cartesian_from_cartesian

*Description:* Create a Cartesian coordinate object from a Cartesian vector.

*Code lines:* 8
18. Source Code Documentation

Containing documentation

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

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.
18.1. Program units

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: 892

module: histories
Description: Defines the history object type.
Code lines: 872
Contained by: file objects.history.F90
Used by: module galacticus_output_star_-
 formation_histories subroutine star_formation_history_-
 formation_histories subroutine star_formation_history_-
 formation_histories subroutine star_formation_history_-
 formation_histories subroutine star_formation_history_-
 metallicity_split_make_history subroutine star_formation_history_-
 metallicity_split_create_metallicity_split subroutine star_formation_history_-
 metallicity_split_output_metallicity_split
subroutine star_formation_history_record_metallicity_split
subroutine star_formation_history_record_null
subroutine output_times_initialize
subroutine galacticus_state_store

subroutine node_component_disk_exponential_create
subroutine node_component_disk_exponential_post_evolve
subroutine node_component_disk_exponential_satellite_merging
subroutine node_component_disk_exponential_star_formation_history_output

module galacticus_nodes

module stellar_population_properties
subroutine stellar_population_properties_history_create_instantaneous
subroutine stellar_population_properties_rates_instantaneous
subroutine stellar_population_properties_scales_instantaneous
subroutine stellar_population_properties_history_create_noninstantaneous
subroutine stellar_population_properties_rates_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: fgs1

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: 166
Contained by: module histories

function: history_add
Description: Add two history objects.
Code lines: 22
Contained by: module histories
subroutine: history_builder
Description: Build a history object from the given XML historyDefinition.
Code lines: 10
Contained by: module histories
Modules used: fox_dom

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: 78
Contained by: module histories
Modules used: arrays_search

subroutine: history_create
Description: Create a history object.
Code lines: 36
Contained by: module histories
Modules used: galacticus_error

subroutine: history_deserialize
Description: Pack history from an array into a history structure.
Code lines: 9
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
18.1. Program units

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: 71
Contained by: module histories
Modules used: fgsl galacticus_error numerical_interpolation

function: history_is_zero
Description: Test whether a history object is all zero.
Code lines: 10
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

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: 15
  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

file: objects.kepler_orbits.F90
  Description: Contains a module which defines an orbit structure for use in GALACTICUS.
  Code lines: 996
module: kepler_orbits

Description: Defines an orbit structure for use in GALACTICUS.

Code lines: 976

Contained by: file objects.kepler_orbits.F90

Used by:
- subroutine galacticus_output_tree_satellite_pericenter
- subroutine scan_for_mergers
- module node_component_satellite_standard
- function satellite_time_until_merging_boylandkolchin2008
- function satellite_time_until_merging_lacey_cole
- function satellite_time_until_merging_wetzel_white
- function satellite_time_until_merging
- function virial_orbital_parameters
- function virial_orbital_parameters_fixed
- function satellite_orbit_equivalent_circular_orbit_radius
- program tests_kepler_orbits
- function hot_halo_ram_pressure_force_font2008_get
- subroutine node_component_satellite_very_simple_create
- function satellite_time_until_merging_jiang2008
- function satellite_time_until_merging_lacey_cole_tormen
- function satellite_time_until_merging_preset
- function virial_orbital_parameters_benson2005
- function virial_orbital_parameters_wetzel2010
- function satellite_orbit_convert_to_current_potential
- subroutine satellite_orbit_pericenter_phase_space_coordinates

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

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.
Function: \texttt{kepler\_orbits\_eccentricity}
\textit{Description:} Return the eccentricity for this orbit.
\textit{Code lines:} 18
\textit{Contained by:} module \texttt{kepler\_orbits}

Subroutine: \texttt{kepler\_orbits\_eccentricity\_set}
\textit{Description:} Sets the tangential velocity to the specified value.
\textit{Code lines:} 13
\textit{Contained by:} module \texttt{kepler\_orbits}

Function: \texttt{kepler\_orbits\_energy}
\textit{Description:} Return the energy for this orbit.
\textit{Code lines:} 20
\textit{Contained by:} module \texttt{kepler\_orbits}
\textit{Modules used:} \texttt{numerical\_constants\_physical}

Subroutine: \texttt{kepler\_orbits\_energy\_set}
\textit{Description:} Sets the tangential velocity to the specified value.
\textit{Code lines:} 13
\textit{Contained by:} module \texttt{kepler\_orbits}

Function: \texttt{kepler\_orbits\_host\_mass}
\textit{Description:} Return the host mass for this orbit.
18.1. Program units

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

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

Subroutine: 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
<table>
<thead>
<tr>
<th>Function/Module</th>
<th>Description</th>
<th>Code Lines</th>
<th>Contains</th>
<th>Used Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>kepler_orbits_radius</td>
<td>Propagate an orbit along its path.</td>
<td>32</td>
<td>module kepler_orbits</td>
<td>galacticus_error, numerical_constants_physical</td>
</tr>
<tr>
<td>kepler_orbits_radius_set</td>
<td>Return the radius for this orbit.</td>
<td>12</td>
<td>module kepler_orbits</td>
<td>galacticus_error</td>
</tr>
<tr>
<td>kepler_orbits_read_raw</td>
<td>Sets the radius to the specified value.</td>
<td>13</td>
<td>module kepler_orbits</td>
<td></td>
</tr>
<tr>
<td>kepler_orbits_reset</td>
<td>Read a keplerOrbit object in binary.</td>
<td>18</td>
<td>module kepler_orbits</td>
<td></td>
</tr>
<tr>
<td>kepler_orbits_semi_major_axis</td>
<td>Reset an orbit to a null state.</td>
<td>20</td>
<td>module kepler_orbits</td>
<td></td>
</tr>
<tr>
<td>kepler_orbits_semi_major_axis_set</td>
<td>Return the semi-major axis for this orbit.</td>
<td>18</td>
<td>module kepler_orbits</td>
<td></td>
</tr>
<tr>
<td>kepler_orbits_specific_reduced_mass</td>
<td>Sets the semi-major axis to the specified value.</td>
<td>13</td>
<td>module kepler_orbits</td>
<td>galacticus_error</td>
</tr>
<tr>
<td>kepler_orbits_velocity_radial</td>
<td>Return the specific reduced mass for this orbit.</td>
<td>12</td>
<td>module kepler_orbits</td>
<td>galacticus_error</td>
</tr>
<tr>
<td>kepler_orbits_velocity_radial</td>
<td>Return the radial velocity for this orbit.</td>
<td>22</td>
<td>module kepler_orbits</td>
<td>galacticus_error</td>
</tr>
</tbody>
</table>
**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**

*Description:* Contains a module which implements the mass distribution class.

*Code lines:* 281

**module: mass_distributions**

*Description:* Implements the mass distribution class.

*Code lines:* 261

*Contained by:* file objects.mass_distributions.F90

*Modules used:* fgsl

*Used by:* subroutine galacticus_state_retrieve subroutine galacticus_state_store

module node_component_spheroid_standard_data

program test_mass_distributions
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_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: 10
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_is_dimensionless
Description: Return true if self is a dimensionless mass distribution.
Code lines: 7
Contained by: 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
Contained by: 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
Contained by: 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
18.1. Program units

**Contained by:** module *mass_distributions*

**Modules used:**
- `coordinates`
- `iso_c_binding`

**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:**
- `coordinates`
- `galacticus_error`

**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:** 49

**Contained by:** module *mass_distributions*

**type:** *massdistributioncylindrical*

**Description:** A cylindrical mass distribution class.

**Code lines:** 4

**Contained by:** module *mass_distributions*

**type:** *massdistribution_spherical*

**Description:** A spherical mass distribution class.

**Code lines:** 15

**Contained by:** module *mass_distributions*

**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:** 1831

**module:** *merger_tree_data_structure*

**Description:** Implements an object to store merger tree data for processing into GALACTICUS’s preferred file format.

**Code lines:** 1811

**Contained by:** file *objects.merger_tree_data.F90*

**Modules used:**
- `iso_varying_string`
- `kind_numbers`
18. Source Code Documentation

Used by: program millennium_merger_tree_file_maker  program simple_merger_tree_file_maker
        subroutine merger_trees_millennium_process  subroutine merger_trees_simple_process
        subroutine merger_tree_write

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: 10
Contained by: module merger_tree_data_structure

subroutine: merger_tree_data_structure_add_metadata
Description: Add a metadatum.
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 metadatum.
Code lines: 10
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: 19
Contained by: module merger_tree_data_structure
Modules used: galacticus_error string_handling

subroutine: merger_tree_data_structure_export_galacticus
Description: Output a set of merger trees to a Galacticus-format HDF5 file.
Code lines: 244
Contained by: module merger_tree_data_structure
18.1. Program units

Modules used:

- hdf5
- io_hdf5
- 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: 278
Contained by: module merger_tree_data_structure
Modules used:

- array_utilities
- galacticus_error
- 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: 222
Contained by: module merger_tree_data_structure
Modules used:

- file_utilities
- galacticus_error
- galacticus_display
- 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: 48
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
18. Source Code Documentation

**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`

**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:** 58
- **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_tree_count`

- **Description:** Set the total number of trees in merger trees.
- **Code lines:** 10

---

912
18.1. Program units

Contained by: module `merger_tree_data_structure`

**subroutine: merger_tree_data_structure_set_tree_indices**
*Description:* Set the merger tree index arrays.
*Code lines:* 29
*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
*Contained by:* module `merger_tree_data_structure`
*Modules used:* `galacticus_error`

**type: mergertreedata**
*Description:* A structure that holds raw merger tree data.
*Code lines:* 153
*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.
module: merger_trees_dump
Description: Implements dumping of the structure of a merger tree to a file for plotting with DOT.
Code lines: 195
Contained by: file objects.merger_trees.dump.F90
Modules used: kind_numbers
Used by: 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: 35507

module: galacticus_nodes
Description: Implements an object hierarchy for nodes in merger trees and all of their constituent physical components.
Code lines: 35486
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
Used by: module accretion_halos subroutine halo_baryonic_accreted_abundances_null_get
18.1. Program units

subroutine halo_baryonic_accreted_chemicals_null_get
subroutine halo_baryonic_accretion_rate_abundances_null_get
function halo_baryonic_accretion_rate_null_get
function halo_baryonic_failed_accretion_rate_null_get
subroutine halo_baryonic_accreted_abundances_simple_get
function halo_baryonic_accreted_mass_simple_get
subroutine halo_baryonic_accretion_rate_chemicals_simple_get
function halo_baryonic_failed_accreted_mass_simple_get
function accretion_disk_jet_power_adaf
function black_hole_spin_up_rate_adaf
function accretion_disk_radiative_efficiency_adaf
function accretion_disk_jet_power
function black_hole_spin_up_rate
function accretion_disk_radiative_efficiency_shakura_sunyaev
function accretion_disk_jet_power_switched
function accretion_disk_switched_adaf_radiative_efficiency_fraction
function black_hole_spin_up_rate_switched
function black_hole_binary_initial_radius_volonteri_2003
module black_hole_binary_initial_radii_tidal_radius
function black_hole_binary_separation_growth_rate_null
module black_hole_fundamentals
module cooling_radii_isothermal
function cooling_ratecole2000
module cooling_rates_white_frenk

subroutine halo_baryonic_accreted_chemicals_null_get
subroutine halo_baryonic_accretion_rate_chemicals_null_get
function halo_baryonic_failed_accreted_mass_null_get
subroutine get_chemical_masses
subroutine halo_baryonic_accreted_abundances_simple_get
function halo_baryonic_accreted_mass_simple_get
subroutine halo_baryonic_accretion_rate_chemicals_simple_get
function accretion_disk_radiative_efficiency
function black_hole_spin_up_rate
function accretion_disk_radiative_efficiency_shakura_sunyaev
function accretion_disk_switched_radiative_efficiency
function black_hole_binary_initial_radius
function black_hole_binary_initial_radius_spheroid_size
function black_hole_binary_separation_growth_rate
function black_hole_binary_separation_growth_rate_stdard
module cooling_radii
module cooling_radii_simple
module cooling_rates
subroutine cooling_rate_modifier_cut-off
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_accretion_histories_wechsler2002
module dark_matter_halos_mass_loss_rates
function dark_matter_halos_mass_loss_rate_vandenbosch
function dark_matter_halo_angular_momentum
subroutine dark_matter_halo_spins_initialize
module halo_spin_distributions
function halo_spin_distribution_lognormal
module dark_matter_profiles
function dark_matter_profile_circular_velocity_isothermal
function dark_matter_profile_density_isothermal
function dark_matter_profile_mass_isothermal
function dark_matter_profile_energy_growth_rate_isothermal
function dark_matter_profile_freefall_radius_increase_rate_isothermal
function dark_matter_profile_freefall_radius_isothermal
function dark_matter_profile_kspace_isothermal
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_accretion_histories_wechsler2002
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_function
module dark_matter_profiles_einasto
module dark_matter_profiles_nfw
function dark_matter_profile_density_isothermal
function dark_matter_profile_energy_growth_rate_isothermal
function dark_matter_profile_freefall_radius_increase_rate_isothermal
function dark_matter_profile_kspace_isothermal
subroutine galacticus_output_halo_model
subroutine galacticus_output_halo_model_property_count
subroutine galacticus_output_tree_main_branch
subroutine galacticus_output_tree_main_branch_property_count
subroutine galacticus_output_most_massive_progenitor
subroutine galacticus_output_most_massive_progenitor_property_count
subroutine galacticus_output_halo_model_names
module galacticus_output_trees_links
subroutine galacticus_output_tree_main_branch_names
module galacticus_output_tree_mass_profiles
subroutine galacticus_output_most_massive_progenitor_names
subroutine galacticus_output_tree_rotation_curve
subroutine galacticus_output_tree_rotation_curve_names
subroutine galacticus_output_tree_rotation_curve_property_count
subroutine galacticus_output_tree_rotation_curve_initialize
module galacticus_output_star_formation_histories
subroutine star_formation_history_create_metallicity_split
subroutine star_formation_history_output_metallicity_split
subroutine star_formation_history_record_metallicity_split
subroutine star_formation_history_create_null
subroutine star_formation_history_output_null
subroutine star_formation_history_record_null
module galacticus_output_trees_velocity_dispersion
subroutine galacticus_output_tree_virial
subroutine galacticus_output_tree_virial_property_count
subroutine get_tree
module hot_halo_density_profile
function hot_halo_density_cored_isothermal_enclosed_mass_get
function hot_halo_density_cored_isothermal_log_slope_get
function hot_halo_profile_rotation_normalization_cored_isothermal_get
module hot_halo_density_cored_isothermal_core_radii_growing_core
function hot_halo_density_null_enclosed_mass_get
function hot_halo_density_cored_isothermal
function hot_halo_density_cored_isothermal_get
function hot_halo_profile_radial_moment_cored_isothermal_get
module hot_halo_density_cored_isothermal_core_radii
function hot_halo_density_cored_isothermal_core_radius_virial_fraction
function hot_halo_density_null_get
function hot_halo_density_null_log_slope_get
function hot_halo_density_null_rotation_normalization_get
module hot_halo_ram_pressure_force_font2008
function hot_halo_ram_pressure_stripping_radius
function hot_halo_ram_pressure_stripping_virial_radius
function hot_halo_temperature_logarithmic_slope_virial_get
module merger_tree_construction

module merger_tree_build

function node_lookup
subroutine merger_tree_smooth_accretion_do
subroutine merger_tree_state_store
function pointed_at_node
module merger_trees_evolve_node
subroutine merger_tree_record_evolution_output
subroutine satellite_merger_process
module merger_trees_evolve_timesteps_template
subroutine merger_tree_mass_accretion_history_output
subroutine merger_tree_structure_output
subroutine merger_tree_prune_hierarchy
subroutine merger_trees_render_dump
subroutine merger_tree_dump

function cnode_component_basic_accretionrate
function cnode_component_basic_time

function cnode_component_blackhole_mass
function cnode_component_blackhole_radialposition
function cnode_component_blackhole_spinseed

function hot_halo_density_null_radial_moment_get
function hot_halo_ram_pressure_force

module hot_halo_ram_pressure_force_null_get
module hot_halo_ram_pressure_stripping_font2008
module hot_halo_temperature_profile

function hot_halo_temperature_virial_get
subroutine merger_tree_build_do_cole2000
subroutine merger_tree_construct_fully_specified
module merger_tree_read
subroutine merger_tree_state_restore

subroutine merger_tree_state_store
subroutine merger_tree_state_walk_tree
module merger_trees_evolve
module merger_tree_timesteps_history
subroutine merger_tree_record_evolution_store
subroutine merger_tree_timestep_simple
subroutine merger_tree_initialize

subroutine merger_tree_monotonic_mass_growth
subroutine merger_tree_prune_branches
subroutine merger_tree_regrid_time
subroutine merger_tree_write
function merger_tree_cladistic_information_content
function cnode_component_basic_mass
function cnode_component_basic_timelastisolated
function cnode_component_blackhole_massseed
function cnode_component_blackhole_spinseed
function cnode_component_blackhole_tripleinteractiontime
function cnode_component_-darkmatterprofile_scale
function cnode_component_-darkmatterprofile_shape
function cnode_component_disk_-angularmomentum
function cnode_component_disk_massgas
function cnode_component_disk_radius
function cnode_component_disk_velocity
function cnode_component_hothalo_-angularmomentum
function cnode_component_hothalo_-outerradius
function cnode_component_hothalo_-outflowedmass
function cnode_component_hothalo_-unaccretedmass
function cnode_component_interoutput_-spheroidealstarformationrate
function cnode_component_satellite_-boundmass
function cnode_component_satellite_-timeofmerging
function cnode_component_spheroid_-halfmassradius
function cnode_component_spheroid_-massstellar
function cnode_component_spheroid_-starformationrate
function cnode_component_spin_-spin
module node_component_basic_non_-evolving
module node_component_black_hole_-simple
module node_component_black_hole_-standard
module node_component_black_hole_-standard
FUNCTION cnode_component_-darkmatterprofile_scalegrowthrate
FUNCTION cnode_component_-darkmatterprofile_shapegrowthrate
FUNCTION cnode_component_disk_-halfmassradius
FUNCTION cnode_component_disk_-massstellar
FUNCTION cnode_component_disk_-starformationrate
FUNCTION cnode_component_hothalo_-outflowedangularmomentum
FUNCTION cnode_component_hothalo_-strippedmass
FUNCTION cnode_component_interoutput_-diskstarformationrate
FUNCTION cnode_component_-mergingstatistics_-galaxymajormergertime
FUNCTION cnode_component_-mergingstatistics_nodeformationtime
FUNCTION cnode_component_-mergingstatistics_nodemajormergertime
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
FUNCTION cnode_component_black_hole_-simple_potential
FUNCTION cnode_component_black_hole_-simple_rotation_curve
FUNCTION cnode_component_black_hole_-simple_rotation_curve_gradient
FUNCTION cnode_component_black_hole_-standard_potential
18.1. Program units

function node_component_black_hole_standard_rotation_curve
module node_component_dark_matter_profile_scale
module node_component_disk_exponential
module node_component Formation_times_cole2000
module node_component_hot_halo_simple
module node_component_inter_output_standard
module node_component_merging_statistics_recent
module node_component_merging_statistics_standard
module node_component_satellite_preset
module node_component_satellite_very_simple
module node_component_spin_preset
module node_component_spin_random
subroutine radiation_set
module radiation_intergalactic_background
subroutine radiation_set_null
function ram_pressure_stripping_mass_loss_rate_disk
function ram_pressure_stripping_mass_loss_rate_disk_null
function satellite_time_until_merging_boylankolchin2008
function satellite_time_until_merging_lacey_cole
function satellite_time_until_merging_wetzel_white
subroutine satellite_merging_mass_movement
subroutine satellite_merging_mass_movement_simple
function satellite_time_until_merging_preset
subroutine satellite_merging_remnant_size_covington2008
subroutine satellite_merging_remnant_size_null
subroutine satellite_merging_remnant_progenitor_properties
subroutine satellite_merging_remnant_progenitor_properties_standard_init
function node_component_black_hole_standard_rotation_curve_gradient
module node_component_dark_matter_profile_scale_shape
module node_component_disk_very_simple
module node_component_hot_halo_standard
module node_component_indices_standard
module node_component_satellite_standard
module node_component_satellite_very_simple
module node_component_spin_random
subroutine radiation_set_cmb
subroutine radiation_igb_file_set
subroutine satellite_merging_mass_movement_baugh2005
subroutine satellite_merging_mass_movement
subroutine satellite_merging_mass_movement_store
subroutine satellite_merging_mass_movement_very_simple
subroutine satellite_merging_remnant_size_cole2000
subroutine satellite_merging_remnant_size
module satellite_merging_remnant_progenitors_properties_cole2000
subroutine satellite_merging_remnant_progenitor_properties
subroutine satellite_merging_remnant_progenitor_properties_standard
function satellite_time_until_merging
function virial_orbital_parameters_benson2005
function virial_orbital_parameters_wetzel2010
module satellite_orbits
module star_formation_feedback_disks_crancey2012
module star_formation_feedback_disks_fixed
module star_formation_feedback_disks_halo_scaling
module star_formation_feedback_spheroids
module star_formation_feedback_expulsion_disks
module star_formation_feedback_expulsion_spheroids
module star_formation_feedback_spheroids
module star_formation_feedback_expulsion_spheroids
module star_formation_feedback_expulsion_spheroids
module star_formation_feedback_spheroids
module star_formation_feedback_expulsion_spheroids
module star_formation_feedback_spheroids
module star_formation_feedback_spheroids
module star_formation_rate_surface_density_disks
module star_formation_rate_surface_density_disks_br
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_rate_surface_density_disks
module star_formation_rate_surface_density_disks
module star_formation_rate_surface_density_disks
module star_formation_rate_surface_density_disks
module star_formation_timescales_disks_dynamical_time
module star_formation_timescale_disks_dynamical_time_initialize
module star_formation_timescale_disks_halo_scaling
module star_formation_timescale_disks_integrated_sd
module star_formation_timescales_spheroids
module star_formation_timescales_spheroids
module star_formation_timescales_spheroids
module star_formation_timescales_spheroids
module star_formation_timescales_spheroids
module stellar_population_properties
module stellar_population_properties
module stellar_population_properties
module dark_matter_halo_biases
function dark_matter_halo_bias_node_press_schechter
function dark_matter_halo_bias_node_tinker2010
program test_prada2011_concentration_dark_energy
program test_zhao2009_dark_energy
18.1. Program units

```
program tests_bug745815
subroutine test_node_task
subroutine testvoidfunc

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: 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
  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: 30
  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
```
18. Source Code Documentation

*Contained by:* module `galacticus_nodes`

**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:** basicnonevolvingmassget
*Description:* Return the `mass` property of the `BasicNonEvolving` component implementation.
*Code lines:* 8
*Contained by:* module `galacticus_nodes`

**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`

**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
18.1. Program units

Contained by: module galacticus_nodes
Modules used: memory_management

function: basicnullbindingdouble0inout
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
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

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
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: basictime
Description: Returns the default value for the time property for the basic component class.
Code lines: 8
Contained by: module galacticus_nodes

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: basictimelastisolatedisgettable
Description: Returns true if the timeLastIsolated property is gettable for the basic 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: 30
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: 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: 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

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.
18.1. Program units

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: blackholeradialpositionisgettable
Description: Returns true if the radialPosition property is gettable for the blackHole component class.
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.
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
Description: Returns the default value for the spinSeed property for the blackHole component class.
Code lines: 8
Contained by: module galacticus_nodes

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: 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

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: 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: 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: 30
Contained by: module galacticus_nodes
Modules used: galacticus_error

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: darkmatterprofilescale
Description: Returns the default value for the scale property for the darkMatterProfile component class.
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 darkMatterProfile 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 darkMatterProfile component class has been attached.
**Code lines:** 5
**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:** 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:** 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

**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: darkmatterprofilescalescalegrowthrateset
Description: Set the scaleGrowthRate property of the DarkMatterProfileScale component implementation.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: darkmatterprofilescalescalerate
Description: Accumulate to the scale property rate of change of the DarkMatterProfileScale component implementation.
Code lines: 10
Contained by: module galacticus_nodes

subroutine: darkmatterprofilescalescalescale
Description: Set the scale property scale of the DarkMatterProfileScale component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: darkmatterprofilescalescaleset
Description: Set the scale property of the DarkMatterProfileScale 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 DarkMatterProfileScaleShape component implementation.
Code lines: 7
Contained by: module galacticus_nodes

function: darkmatterprofilescaleshapeshapeget
Description: Return the shape property of the DarkMatterProfileScaleShape component implementation.
Code lines: 8
Contained by: module galacticus_nodes

function: darkmatterprofilescaleshapeshapegrowthrateget
Description: Return the shapeGrowthRate property of the DarkMatterProfileScaleShape component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: darkmatterprofilescaleshapeshapegrowthrateset
Description: Set the shapeGrowthRate property of the DarkMatterProfileScaleShape 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 DarkMatterProfileScaleShape component implementation.  
Code lines: 10  
Contained by: module galacticus_nodes

subroutine: darkmatterprofilescaleshapeshapescale  
Description: Set the shape property scale of the DarkMatterProfileScaleShape component implementation.  
Code lines: 8  
Contained by: module galacticus_nodes

subroutine: darkmatterprofilescaleshapeshapeset  
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: darkmatterprofileshapegrowthrate  
Description: Returns the default value for the shapeGrowthRate property for the darkMatterProfile component class.  
Code lines: 8  
Contained by: module galacticus_nodes

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: 219  
Contained by: module galacticus_nodes  
Modules used: memory_management

subroutine: deserializefromarrayscales
18.1. Program units

Description: Deserialize scales from array.
Code lines: 219
Contained by: module galacticus_nodes
Modules used: memory_management

**subroutine: deserializefromarrayvalues**
Description: Deserialize values from array.
Code lines: 219
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: 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: 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: 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: diskcreatebyinterrupt
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
Modules used: galacticus_display iso_varying_string string_handling

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.
18.1. Program units

**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

**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

*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

*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: 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:* 11

*Contained by:* module galacticus_nodes

**Function: diskexponentialluminositiesstellarget**
18.1. Program units

**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:** 35

**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.
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.
Code lines: 10
Contained by: module galacticus_nodes

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
18.1. Program units

**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`

**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`

**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.
18. Source Code Documentation

```
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.
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
Modules used: memory_management

function: diskget
Description: Returns the disk component of self.
Code lines: 30
Contained by: module galacticus_nodes
Modules used: galacticus_error

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: 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: diskisinitializedisgettable
```

944
18.1. Program units

**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:** 14
**Contained by:** module `galacticus_nodes`
**Modules used:** `stellar_population_properties_luminosities`

**function: diskluminositiesstellarisgettable**

**Description:** Returns true if the `luminositiesStellar` property is gettable for the disk component class.

**Code lines:** 7
**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: 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.

**Code lines:** 8
**Contained by:** module `galacticus_nodes`

**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 type abundances.

**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: disknullbindingdouble1inout
   Description: A null get function for rank 1 double precisions.
   Code lines: 7
   Contained by: module galacticus_nodes

function: disknullbindinghistory0inout
   Description: A null get function for rank 0 typehistorys.
   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 typeabundancess.
   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: disknullbindingratedouble1inout
   Description: A null rate function for rank 1 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
18.1. Program units

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: disknullbindingsetabundances0inout
Description:  A null set function for rank 0 type abundances.
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: disknullbindingsetdouble1inout
Description:  A null set function for rank 1 double precisions.
Code lines:  7
Contained by:  module galacticus_nodes

subroutine: disknullbindingsethistory0inout
Description:  A null set function for rank 0 type histories.
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

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: 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.
function: diskstarformationhistoryisgettable
Description: Returns true if the starFormationHistory property is gettable for the disk component class.

function: diskstarformationrate
Description: Returns the default value for the starFormationRate property for the disk component class.

subroutine: diskstarformationrategetfunction
Description: Set the function to be used for get of the starFormationRate property of the disk component class.

function: diskstarformationrategetisattached
Description: Return true if the deferred function used to get the starFormationRate property of the disk component class has been attached.

function: diskstarformationrateisgettable
Description: Returns true if the starFormationRate property is gettable for the disk component class.

function: diskstellarpropertieshistory
Description: Returns the default value for the stellarPropertiesHistory property for the disk component class.

function: diskstellarpropertieshistoryisgettable
Description: Returns true if the stellarPropertiesHistory property is gettable for the disk component class.

function: diskvelocity
Description: Returns the default value for the velocity property for the disk component class.

function: diskvelocityisgettable
18.1. Program units

**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.
**Code lines:** 7
**Contained by:** module galacticus_nodes

**function:** diskverysimplemassgasget
**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_error
**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:** diskverysimplemassstellarcound
**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:** diskverysimplemassstellartarget
**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
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: 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: formationtimeformationtimeisgettable
Description: Returns true if the formationTime property is gettable for the formationTime component class.
Code lines: 6
18.1. Program units

Contains by: module galacticus_nodes

function: formationtimeget
Description: Returns the formationTime component of self.
Code lines: 30
Contains by: module galacticus_nodes
Modules used: galacticus_error

function: formationtimenullbindingdouble0inout
Description: A null get function for rank 0 double precisions.
Code lines: 7
Contains by: module galacticus_nodes

function: formationtimenullbindinginteger0in
Description: A null get function for rank 0 integers.
Code lines: 7
Contains by: module galacticus_nodes

subroutine: formationtimenullbindingratedouble0inout
Description: A null rate function for rank 0 double precisions.
Code lines: 9
Contains by: module galacticus_nodes

subroutine: formationtimenullbindingrateinteger0in
Description: A null rate function for rank 0 integers.
Code lines: 9
Contains by: module galacticus_nodes

subroutine: formationtimenullbindingsetdouble0inout
Description: A null set function for rank 0 double precisions.
Code lines: 7
Contains by: module galacticus_nodes

subroutine: formationtimenullbindingsetinteger0in
Description: A null set function for rank 0 integers.
Code lines: 7
Contains by: module galacticus_nodes

subroutine: galacticus_nodes_finalize
Description: Finalize the GALACTICUS object system.
Code lines: 19
Contains by: module galacticus_nodes

subroutine: galacticus_nodes_initialize
Description: Initialize the GALACTICUS object system.
Code lines: 419
Contains 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**: genericnullbinding abundances0inout  
**Description**: A null get function for rank 0 type abundances.  
**Code lines**: 7  
**Contained by**: module galacticus_nodes

**function**: genericnullbinding double0inout  
**Description**: A null get function for rank 0 double precisions.  
**Code lines**: 7  
**Contained by**: module galacticus_nodes

**subroutine**: genericnullbinding rate abundances0inout  
**Description**: A null rate function for rank 0 type abundances.  
**Code lines**: 9  
**Contained by**: module galacticus_nodes

**subroutine**: genericnullbinding rated double0inout  
**Description**: A null rate function for rank 0 double precisions.  
**Code lines**: 9  
**Contained by**: module galacticus_nodes

**subroutine**: genericnullbinding set abundances0inout  
**Description**: A null set function for rank 0 type abundances.  
**Code lines**: 7  
**Contained by**: module galacticus_nodes

**subroutine**: genericnullbinding set double0inout  
**Description**: A null set function for rank 0 double precisions.  
**Code lines**: 7  
**Contained by**: module galacticus_nodes

**function**: hothalo abundances  
**Description**: Returns the default value for the abundances property for the hotHalo component class.  
**Code lines**: 8  
**Contained by**: module galacticus_nodes

**function**: hothalo abundances is gettable  
**Description**: Returns true if the abundances property is gettable for the hotHalo component class.  
**Code lines**: 6  
**Contained by**: module galacticus_nodes

**function**: hothalo angular momentum  
**Description**: Returns the default value for the angularMomentum property for the hotHalo component class.  
**Code lines**: 8
18.1. Program units

function: hothaloangularmomentumisgettable
Description: Returns true if the angularMomentum property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

function: hothalochemicals
Description: Returns the default value for the chemicals property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalochemicalsisgettable
Description: Returns true if the chemicals property is gettable for the hotHalo component class.
Code lines: 6
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: 8
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 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: 30
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: hothaloheatsourceisgettable
Description: Returns true if the heatSource property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: hothaloheatsourceratefunction
Description: Set the function to be used for rate of the heatSource property of the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothaloheatsourcerateisattached
Description: Return true if the deferred function used to rate the heatSource property of the hotHalo component class has been attached.
Code lines: 5
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: 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
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: hothalohothalocoolingabundancesrateisattached
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: hothalohothalocoolingangularmomentum
Description: Returns the default value for the hotHaloCoolingAngularMomentum property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalohothalocoolingangularmomentumisgettable
Description: Returns true if the hotHaloCoolingAngularMomentum property is gettable for the hotHalo component class.
18.1. Program units

Code lines: 6
Contained by: module galacticus_nodes

subroutine: hothalohothalocoolingangularmomentumratefunction
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: hothalohothalocoolingangularmomentumrateisattached
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: hothalohothalocoolingmass
Description: Returns the default value for the hotHaloCoolingMass property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

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: hothalomass
Description: Returns the default value for the mass property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalomassisgettable
Description: Returns true if the mass property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

function: hothalomasssink
18. Source Code Documentation

**Description:** Returns the default value for the massSink property for the hotHalo component class.

**Code lines:** 8

**Contained by:** module galacticus_nodes

**function: hothalomasssinkisgettable**

**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

**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: 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

function: hothaloouterradius
Description: Returns the default value for the outerRadius property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: hothaloouterradiusgetfunction
Description: Set the function to be used for get of the outerRadius property of the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothaloouterradiusgetisattached
Description: Return true if the deferred function used to get the outerRadius property of the hotHalo component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes

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
Description: Returns the default value for the outflowedAbundances property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalooutflowedabundancesisgettable
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: 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: hothalooutflowedmassisgettable
Description: Returns true if the outflowedMass property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

function: hothalooutflowingabundances
Description: Returns the default value for the outflowingAbundances property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalooutflowingabundancesisgettable
Description: Returns true if the outflowingAbundances property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: hothalooutflowingabundancesratefunction
Description: Set the function to be used for rate of the outflowingAbundances property of the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalooutflowingabundancesrateisattached
Description: Return true if the deferred function used to rate the outflowingAbundances property of the hotHalo component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes
function: hothalooutflowingangularmomentum
Description: Returns the default value for the outflowingAngularMomentum property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalooutflowingangularmomentumisgettable
Description: Returns true if the outflowingAngularMomentum property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: hothalooutflowingangularmomentumratefunction
Description: Set the function to be used for rate of the outflowingAngularMomentum property of the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalooutflowingangularmomentumrateisattached
Description: Return true if the deferred function used to rate the outflowingAngularMomentum property of the hotHalo component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes

function: hothalooutflowingmass
Description: Returns the default value for the outflowingMass property for the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalooutflowingmassisgettable
Description: Returns true if the outflowingMass property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: hothalooutflowingmassratefunction
Description: Set the function to be used for rate of the outflowingMass property of the hotHalo component class.
Code lines: 8
Contained by: module galacticus_nodes

function: hothalooutflowingmassrateisattached
Description: Return true if the deferred function used to rate the outflowingMass property of the hotHalo component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes

function: hothalooutflowingmassrateisattached
Description: Return a count of the number of scalar properties in the abundances property of the hotHaloStandard component implementation.
18. Source Code Documentation

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: hothalostandardabundancesset
Description: Set the abundances property scale 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
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.
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
Modules used: memory_management

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: hothalostandardheatsourceerate
Description: Set the rate of the heatSource property of the HotHaloStandard component using a deferred function.

Code lines: 10
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: 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`

**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`

**function: hothalostandardouterradiusgetvalue**
Description: Return the `outerRadius` property of the `HotHaloStandard` component implementation.
Code lines: 8
Contained by: `module galacticus_nodes`

**subroutine: hothalostandardouterradiusrate**
18.1. Program units

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
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
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.
18.1. Program units

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: 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: 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

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
18. Source Code Documentation

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
subsection: hothalostandardunaccretedmassset
Description: Set the unaccretedMass property of the HotHaloStandard component implementation.
Code lines: 11
Contained by: module galacticus_nodes

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: hothalostrippedabundancesisgettable
Description: Returns true if the strippedAbundances 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: hothalostrippedmassisgettable
Description: Returns true if the strippedMass 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: hothalounaccretedmassisgettable
Description: Returns true if the unaccretedMass property is gettable for the hotHalo component class.
Code lines: 6
Contained by: module galacticus_nodes

subsection: hothaloverysimplehothalooolcoolingmassrate
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
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

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

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: 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
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: 30
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.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: indicesstandardbranchtipset
Description: Set the branchTip property of the IndicesStandard component implementation.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

function: interoutputcountlinked
Description: Returns the number of interOutput components in self.
Code lines: 19
Contained by: module galacticus_nodes

subroutine: interoutputcreatebyinterrupt
Description: Create the interOutput component of self via an interrupt.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: interoutputcreatelinked
Description: Create the interOutput component of self.
Code lines: 29
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: interoutputdestroylinked
Description: Destroy the interOutput component of self.
Code lines: 13
Contained by: module galacticus_nodes

function: interoutputdiskstarformationrate
Description: Returns the default value for the diskStarFormationRate property for the interOutput component class.
Code lines: 12
Contained by: module galacticus_nodes

function: interoutputdiskstarformationrateisgettable
Description: Returns true if the diskStarFormationRate property is gettable for the interOutput component class.
Code lines: 7
Contained by: module galacticus_nodes

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
18.1. Program units

Contained by: module galacticus_nodes
Modules used: galacticus_error

function: interoutputget
Description: Returns the interOutput component of self.
Code lines: 30
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: 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

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
Modules used: memory_management

function: mapcomponentsdouble0
Description: Map a scalar double function over components with a specified reduction.
Code lines: 159
Contained by: module galacticus_nodes

subroutine: mapcomponentsvoid
Description: Map a void function over components.
Code lines: 73
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

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
18. Source Code Documentation

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: 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
Description: Returns the mergingStatistics component of self.
Code lines: 30
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: 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: 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: 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.
Code lines: 9
Contained by: module galacticus_nodes
18. Source Code Documentation

**Description:** A null rate function for rank 1 integers.

**Code lines:** 9

**Contained by:** module `galacticus_nodes`

**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:** `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`

**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: mergingstatisticsstandardnodemajormergertimeset
Description: Set the nodeMajorMergerTime property of the MergingStatisticsStandard component implementation.
Code lines: 9
module galacticus_nodes

subroutine node_component_assign
Description: Assign a node component to another node component.
Code lines: 248
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_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: 69
Contained by: module galacticus_nodes
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.
18.1. Program units

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:* 47
*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
*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_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_ode_step_scales_init
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
18.1. Program units

**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.

**Code lines:** 10

**Contained by:** module `galacticus_nodes`

**Modules used:** `fox_dom` `galacticus_error` `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.
18. Source Code Documentation

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_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
### 18.1. Program units

**subroutine**: node_component_basicnull_ode_step_scales_init  
*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_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_ode_step_scales_init
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.
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_blackhole_builder
  Description: Build a generic blackHole component.
  Code lines: 10
  Contained by: module galacticus_nodes
  Modules used: fox_dom galacticus_error

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_initilizer
  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: 69
  Contained by: module galacticus_nodes
  Modules used: galacticus_error

subroutine: node_component_blackhole_output
  Description: Output ptoperties 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
18.1. Program units

**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

*Contained by:* 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
18. Source Code Documentation

`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_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_ode_step_scales_init`

*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`
18.1. Program units

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
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 galacticus_error memory_management

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.
18. Source Code Documentation

**subroutine: node_component_blackholesimple_destroy**
*Description:* Destroy a simple implementation of the blackHole component.
*Code lines:* 11
*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:* 7
*Contained by:* module `galacticus_nodes`
*Modules used:* `galacticus_display` `iso_varying_string` `string_handling`

**subroutine: node_component_blackholesimple_dump_raw**
*Description:* Dump the contents of a simple implementation of the blackHole component in binary.
*Code lines:* 16
*Contained by:* module `galacticus_nodes`
*Modules used:* `galacticus_display` `iso_varying_string` `string_handling`

**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_ode_step_scales_init**
*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_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
18.1. Program units

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_ode_step_scales_init
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
Contained by: module galacticus_nodes

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: 69
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_darkmatterprofile_output
Description: Output properties for a darkMatterProfile component.
Code lines: 19
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
Contained by: 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: 47
Contained by: module galacticus_nodes
18.1. Program units

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`
*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_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_ode_step_scales_init
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.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilenull_output_names
Description: Establish property names for a null implementation of the darkMatterProfile component.
Code lines: 11
Contained by: module galacticus_nodes

subroutine: node_component_darkmatterprofilenull_read_raw
18.1. Program units

**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.

**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.
18. Source Code Documentation

**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_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

**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_ode_step_scales_init

**Description:** Initialize scales in a scale implementation of the darkMatterProfile component for an ODE solver step.
18.1. Program units

```
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_darkmatterprofilescale_type
Description: Returns the type for the scale 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.
```
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

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_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
18.1. Program units

**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_ode_step_scales_init`

**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
function: node_component_darkmatterprofilescaleshape_type
Description: Returns the type for the scaleShape implementation of the darkMatterProfile component.
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 tree node 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: 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

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: 69
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_disk_output
Description: Output properties for a disk component.
Code lines: 49
18.1. Program units

**Contained by:** module galacticus_nodes
**Modules used:** stellar_population_properties_-luminosities

**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_diskexponential_builder
**Description:** Build a exponential implementation of the disk component.
**Code lines:** 84
**Contained by:** module galacticus_nodes
**Modules used:** fox_dom galacticus_error memory_management

**function:** node_component_diskexponential_count
**Description:** Return a count of the serialization of a exponential implementation of the disk component.
**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:** 32
**Contained by:** module galacticus_nodes

**subroutine:** node_component_diskexponential_deserialize_scales
**Description:** Serialize scales of a exponential implementation of the disk component.
**Code lines:** 32
**Contained by:** module galacticus_nodes

**subroutine:** node_component_diskexponential_deserialize_values

1003
**18. Source Code Documentation**

*Description:* Serialize values of a exponential implementation of the disk component.
*Code lines:* 32
*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:* 55
*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:* 22
*Contained by:* module `galacticus_nodes`

**subroutine: node_component_diskexponential_initializor**
*Description:* Initialize a exponential implementation of the disk component.
*Code lines:* 22
*Contained by:* module `galacticus_nodes`
*Modules used:* `memory_management`, `stellar_population_properties_-luminosities`

**function: node_component_management_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_step_scales_init**
*Description:* Initialize scales in a exponential implementation of the disk component for an ODE solver step.
18.1. Program units

**Code lines:** 13  
**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_diskexponential_output_count`  
**Description:** Increment output property count for an exponential implementation of the disk component.  
**Code lines:** 21  
**Contained by:** module `galacticus_nodes`  
**Modules used:** `stellar_population_properties_-luminosities`

**subroutine:** `node_component_diskexponential_output_names`  
**Description:** Establish property names for an exponential implementation of the disk component.  
**Code lines:** 115  
**Contained by:** module `galacticus_nodes`  
**Modules used:** `stellar_population_properties_-luminosities`

**subroutine:** `node_component_diskexponential_read_raw`  
**Description:** Read the contents of an exponential implementation of the disk component in binary.  
**Code lines:** 26  
**Contained by:** module `galacticus_nodes`  
**Modules used:** `memory_management`

**subroutine:** `node_component_diskexponential_serialize_rates`  
**Description:** Serialize rates of an exponential implementation of the disk component.  
**Code lines:** 32  
**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_diskexponential_serialize_scales`  
**Description:** Serialize scales of an exponential implementation of the disk component.  
**Code lines:** 32  
**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_diskexponential_serialize_values`  
**Description:** Serialize values of an exponential implementation of the disk component.  
**Code lines:** 32  
**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_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
18.1. Program units

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_ode_step_scales_init
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_serialized_rates
Description: Serialize rates of a null implementation of the disk component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_disknull_serialized_scales
Description: Serialize scales of a null implementation of the disk component.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_disknull_serialized_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

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
18.1. Program units

Contained by: module galacticus_nodes

subroutine: node_component_diskverysimple_initializer
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.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: node_component_diskverysimple_ode_step_scales_init
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
18. Source Code Documentation

**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

**subroutine:** node_component_dump_raw_null
**Description:** Dump a generic tree node component in binary.
**Code lines:** 7
**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
18.1. Program units

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: 55
Contained by: module galacticus_nodes
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.
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_initializor
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
18.1. Program units

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

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_ode_step_scales_init
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
18. Source Code Documentation

*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:* Build a null implementation of the formationTime component.
*Code lines:* 10
*Contained by:* module galacticus_nodes
*Modules used:* fox_dom, galacticus_error, memory_management

**function: node_component_formationtimenull_count**
*Description:* Return a count of the serialization of a null implementation of the formationTime component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine: node_component_formationtimenull_deserialize_rates**
*Description:* Serialize rates of a null implementation of the formationTime component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine: node_component_formationtimenull_deserialize_scales**
*Description:* Serialize scales of a null implementation of the formationTime component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine: node_component_formationtimenull_deserialize_values**
*Description:* Serialize values of a null implementation of the formationTime component.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine: node_component_formationtimenull_destroy**
*Description:* Destroy a null implementation of the formationTime component.
*Code lines:* 7
*Contained by:* module galacticus_nodes
*Modules used:* memory_management

**subroutine: node_component_formationtimenull_dump**
*Description:* Dump the contents of a null implementation of the formationTime component.
*Code lines:* 9
*Contained by:* module galacticus_nodes
*Modules used:* galacticus_display, iso_varying_string, string_handling
subroutine: node_component_formationtimenull_dump_raw
Description: Dump the contents of a null implementation of the formationTime component in binary.
Code lines: 7
Contained by: module galacticus_nodes

subroutine: node_component_formationtimenull_initializer
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 formation-Time 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_ode_step_scales_init
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.
18. Source Code Documentation

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_hothalo_builder
  
  *Description*: Build a generic hotHalo component.
  
  *Code lines*: 10
  
  *Contained by*: module galacticus_nodes
  
  *Modules used*: fox_dom galacticus_error

**subroutine**: node_component_hothalo_destroy
  
  *Description*: Destroys a hotHalo component.
  
  *Code lines*: 7
  
  *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: 69
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_hothalo_output
Description: Output properties for a hotHalo component.
Code lines: 34
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: 47
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_hothalonull_builder
Description: Build a null implementation of the hotHalo component.
Code lines: 10
18. Source Code Documentation

**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
*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 iso_varying_string string_handling

**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_initializor

*Description:* Initialize a null implementation of the hotHalo component.
*Code lines:* 7
*Contained by:* module galacticus_nodes
*Modules used:* memory_management
18.1. Program units

**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_ode_step_scales_init

*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.

*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_hothalostandard_builder
Description: Build a standard implementation of the hotHalo component.
Code lines: 79
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: 50
Contained by: module galacticus_nodes
18.1. Program units

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: 18
- Contained by: module galacticus_nodes

subroutine: node_component_hothalostandard_initializor
- Description: Initialize a standard implementation of the hotHalo component.
- Code lines: 18
- 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_ode_step_scales_init
- 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
18. Source Code Documentation

**Description:** Read the contents of a standard implementation of the hotHalo component in binary.

**Code lines:** 19

**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_hothaloversimple_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_hothaloversimple_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_hothaloversimple_deserialize_rates`

**Description:** Serialize rates of a verySimple implementation of the hotHalo component.

**Code lines:** 11

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_hothaloversimple_deserialize_scales`

**Description:** Serialize scales of a verySimple implementation of the hotHalo component.

**Code lines:** 11

**Contained by:** module `galacticus_nodes`

**subroutine:** `node_component_hothaloversimple_deserialize_values`
18.1. Program units

**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_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*

**subroutine:** node_component_hothaloverysimple_ode_step_scales_init

**Description:** Initialize scales in a verySimple implementation of the hotHalo component for an ODE solver step.

**Code lines:** 6
18. Source Code Documentation

**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

**subroutine:** node_component_hothaloverysimple_output_names
*Description:* Establish property names for a verySimple implementation of the hotHalo component.
*Code lines:* 24

**subroutine:** node_component_hothaloverysimple_read_raw
*Description:* Read the contents of a verySimple implementation of the hotHalo component in binary.
*Code lines:* 9
*Modules used:* `memory_management`

**subroutine:** node_component_hothaloverysimple_serialize_rates
*Description:* Serialize rates of a verySimple implementation of the hotHalo component.
*Code lines:* 11

**subroutine:** node_component_hothaloverysimple_serialize_scales
*Description:* Serialize scales of a verySimple implementation of the hotHalo component.
*Code lines:* 11

**subroutine:** node_component_hothaloverysimple_serialize_values
*Description:* Serialize values of a verySimple implementation of the hotHalo component.
*Code lines:* 11

**function:** node_component_hothaloverysimple_type
*Description:* Returns the type for the verySimple implementation of the hotHalo component.
*Code lines:* 8

**subroutine:** node_component_indices_builder
*Description:* Build a generic indices component.
*Code lines:* 10
*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_initializer
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: 55
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
Contained by: module galacticus_nodes

subroutine: node_component_indicesnull_builder
Description: Build a null implementation of the indices component.
Code lines: 10
18. Source Code Documentation

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_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_ode_step_scales_init
  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.
  Code lines: 7
  Contained by: module galacticus_nodes

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, memory_management, galacticus_error

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

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
18.1. Program units

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_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_ode_step_scales_init
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
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
18. Source Code Documentation

**Source Code Documentation**

**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

**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: 55
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 galacticus_error memory_management

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
18. Source Code Documentation

**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_initializor`

*Description:* Initialize a null implementation of the interOutput component.

*Code lines:* 7

**Contained by:** module `galacticus_nodes`

**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`


18.1. Program units

**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_ode_step_scales_init

**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.

**Code lines:** 7

**Contained by:** module galacticus_nodes

**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.
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
Modules used: fox_dom memory_management galacticus_error

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.
Code lines: 19
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

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_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_ode_step_scales_init
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
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
18. Source Code Documentation

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_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
Modules used: galacticus_display iso_varying_string

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: 69
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 galacticus_error memory_management

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
18. Source Code Documentation

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_initializor
*Description:* Initialize a null implementation of the mergingStatistics component.
*Code lines:* 7
*Contained by:* module galacticus_nodes
*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_ode_step_scales_init
*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_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
18.1. Program units

*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_ode_step_scales_init`  
*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`  
*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.
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
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
18.1. Program units

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_scales_init
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
  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

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 initializes_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.
18.1. Program units

**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

**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_initializer**
*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:* 55
*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.
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
  Contained by: module galacticus_nodes
  Modules used: fox_dom galacticus_error memory_management

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

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
18.1. Program units

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_initializer
Description: Initialize a null implementation of the position component.
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_step_scales_init
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
18. Source Code Documentation

**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.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**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
   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_initializer
   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_ode_step_scales_init
   Description: Initialize scales in a preset implementation of the position component for an ODE solver step.
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
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
  Contained by: module galacticus_nodes

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

subroutine: node_component_satellite_initializor
  Description: Initialize a generic satellite component.
  Code lines: 8
  Contained by: module galacticus_nodes

subroutine: node_component_satellite_move
  Description: Move instances of the satellite component, from one node to another.
  Code lines: 83
  Contained by: module galacticus_nodes

subroutine: node_component_satellite_output
  Description: Output properties for a satellite component.
  Code lines: 32
  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
subroutine: node_component_satellite_remove
Description: Removes an instance of the satellite component, shifting other instances to keep the array contiguous.
Code lines: 55
Contained by: module galacticus_nodes
Modules used: galacticus_error

function: node_component_satellite_type
Description: Returns the type for the satellite component.
Code lines: 8
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
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
18.1. Program units

```
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_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
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_ode_step_scales_init
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.
```
18. Source Code Documentation

Code lines: 8
Contained by: module galacticus_nodes
Modules used: memory_management

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_satellitepreset_builder
Description: Build a preset implementation of the satellite component.
Code lines: 31
Contained by: module galacticus_nodes
Modules used: fox_dom    galacticus_error
                memory_management

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
18.1. Program units

**subroutine**: node_component_satellitepreset_destroy

*Description*: Destroy a preset implementation of the satellite component.

*Code lines*: 9

*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*: 24

*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*: 10

*Contained by*: module galacticus_nodes

**subroutine**: node_component_satellitepreset_initializor

*Description*: Initialize a preset implementation of the satellite component.

*Code lines*: 10

*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_ode_step_scales_init

*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
18. Source Code Documentation

**Description:** Increment output property count for a preset implementation of the satellite component.
**Code lines:** 10
**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:** 37
**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:** 11
**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.
18.1. Program units

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_initializor
Description: Initialize a standard implementation of the satellite component.
Code lines: 12
Contained by: module galacticus_nodes
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_ode_step_scales_init
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_satellitestandardSerialize_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.
Code lines: 13
Contained by: module galacticus_nodes

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
18.1. Program units

**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.
- **Code lines:** 16
- **Contained by:** module galacticus_nodes
- **Modules used:** galacticus_display, iso_varying_string, string_handling

**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_initializor
- **Description:** Initialize a verySimple implementation of the satellite component.
18. Source Code Documentation

Code lines: 8  
Contained by: module galacticus_nodes  
Modules used: memory_management

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  
Modules used: iso_varying_string

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_ode_step_scales_init  
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

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

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: 55
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_spheroid_output
Description: Output properties for a spheroid component.
Code lines: 43
Contained by: module galacticus_nodes
Modules used: stellar_population_properties_luminosities

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_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 memory_management galacticus_error

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
18.1. Program units

**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_initializor
*Description*: Initialize a null implementation of the spheroid component.
*Code lines*: 7
*Contained by*: module galacticus_nodes
*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
*Contained by*: module galacticus_nodes
18. Source Code Documentation

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
Contained by: module galacticus_nodes

subroutine: node_component_spheroidnull_ode_step_scales_init
Description: Initialize scales in a null implementation of the spheroid component for an ODE solver step.
Code lines: 5
Contained 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
Contained 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
Contained 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
Contained 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
Contained by: module galacticus_nodes

subroutine: node_component_spheroidnull_serialize_scales
Description: Serialize scales of a null implementation of the spheroid component.
Code lines: 7
Contained by: module galacticus_nodes

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
18.1. Program units

Description: Build a standard implementation of the spheroid component.
Code lines: 84
Contained by: module galacticus_nodes
Modules used: fox_dom memory_management galacticus_error

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: 32
Contained by: module galacticus_nodes

subroutine: node_component_spheroidstandard_deserialize_scales
Description: Serialize scales of a standard implementation of the spheroid component.
Code lines: 32
Contained by: module galacticus_nodes

subroutine: node_component_spheroidstandard_deserialize_values
Description: Serialize values of a standard implementation of the spheroid component.
Code lines: 32
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: 55
Contained by: module galacticus_nodes
Modules used: galacticus_display iso_varying_string string_handling

subroutine: node_component_spheroidstandard_dump_raw
Description: Dump the contents of a standard implementation of the spheroid component in binary.
Code lines: 22
Contained by: module galacticus_nodes

subroutine: node_component_spheroidstandard_initializor
Description: Initialize a standard implementation of the spheroid component.
Code lines: 22
Contained by: module galacticus_nodes
Modules used: memory_management stellar_population_properties_-luminosities
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_ode_step_scales_init
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: 21
Contained by: module galacticus_nodes
Modules used: stellar_population_properties_luminosities

subroutine: node_component_spheroidstandard_output_names
Description: Establish property names for a standard implementation of the spheroid component.
Code lines: 115
Contained by: module galacticus_nodes
Modules used: stellar_population_properties_luminosities

subroutine: node_component_spheroidstandard_read_raw
Description: Read the contents of a standard implementation of the spheroid component in binary.
Code lines: 26
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: 32
Contained by: module galacticus_nodes
subroutine: node_component_spheroidstandard_serialize_scales
Description: Serialize scales of a standard implementation of the spheroid component.
Code lines: 32
Contained by: module galacticus_nodes

subroutine: node_component_spheroidstandard_serialize_values
Description: Serialize values of a standard implementation of the spheroid component.
Code lines: 32
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.
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: 69
Contained by: module galacticus_nodes
Modules used: galacticus_error

subroutine: node_component_spin_output
Description: Output properties for a spin component.
Code lines: 19
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: 47
   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 galacticus_error memory_management

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_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

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_ode_step_scales_init
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.
18. Source Code Documentation

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

**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_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
18.1. Program units

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_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
**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_ode_step_scales_init`  
**Description:** Initialize scales in a preset implementation of the spin component for an ODE solver step.  
**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`
18.1. Program units

Modules used: 
- fox_dom
- memory_management
- galacticus_error

**function: node_component_spinrandom_count**
- **Description:** Return a count of the serialization of a random implementation of the spin component.
- **Code lines:** 8
- **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
- **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_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.
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: 6
Contained by: module galacticus_nodes

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: 14
Contained by: module galacticus_nodes

subroutine: node_component_spinrandom_ode_step_scales_init
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
18.1. Program units

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: 106
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: 261
Contained by: module galacticus_nodes

subroutine: node_output
Code lines: 76
Contained by: module galacticus_nodes

subroutine: node_output_count
Description: Increment the count of properties to output for this node.
Code lines: 74
Contained by: module galacticus_nodes

subroutine: node_output_names
Description: Establish the names of properties to output for this node.
Code lines: 76
Contained by: module galacticus_nodes

function: node_property_name_from_index
Description: Return the name of a property given its index.
Code lines: 174
Contained by: module galacticus_nodes

Modules used: iso_varying_string

subroutine: node_read_raw
Description: Dump node content in binary.
Code lines: 379
Contained by: module galacticus_nodes
18. Source Code Documentation

**type: nodecomponent**

*Description:* A class for components in nodes.
*Code lines:* 197
*Contained by:* module galacticus_nodes

**type: nodecomponentbasic**

*Description:* Type for the basic component class.
*Code lines:* 202
*Contained by:* module galacticus_nodes

**type: nodecomponentbasicnonevolving**

*Description:* Class for the nonEvolving implementation of the basic component.
*Code lines:* 80
*Contained by:* module galacticus_nodes

**type: nodecomponentbasicnull**

*Description:* Class for the null implementation of the basic component.
*Code lines:* 30
*Contained by:* module galacticus_nodes

**type: nodecomponentbasicstandard**

*Description:* Class for the standard implementation of the basic component.
*Code lines:* 100
*Contained by:* module galacticus_nodes

**type: nodecomponentblackhole**

*Description:* Type for the blackHole component class.
*Code lines:* 265
*Contained by:* module galacticus_nodes

**type: nodecomponentblackholenull**

*Description:* Class for the null implementation of the blackHole component.
*Code lines:* 30
*Contained by:* module galacticus_nodes

**type: nodecomponentblackholesimple**

*Description:* Class for the simple implementation of the blackHole component.
*Code lines:* 56
*Contained by:* module galacticus_nodes

**type: nodecomponentblackholestandard**

*Description:* Class for the standard implementation of the blackHole component.
*Code lines:* 110
*Contained by:* module galacticus_nodes

**type: nodecomponentdarkmatterprofile**

*Description:* Type for the darkMatterProfile component class.
*Code lines:* 216
18.1. Program units

Contained by: module galacticus_nodes

type: nodecomponentdarkmatterprofilenull
Description: Class for the null implementation of the darkMatterProfile component.
Code lines: 30
Contained by: module galacticus_nodes

type: nodecomponentdarkmatterprofilescale
Description: Class for the scale implementation of the darkMatterProfile component.
Code lines: 70
Contained by: module galacticus_nodes

type: nodecomponentdarkmatterprofilescaleshape
Description: Class for the scaleShape implementation of the darkMatterProfile component.
Code lines: 69
Contained by: module galacticus_nodes

type: nodecomponentdisk
Description: Type for the disk component class.
Code lines: 580
Contained by: module galacticus_nodes

type: nodecomponentdiskexponential
Description: Class for the exponential implementation of the disk component.
Code lines: 262
Contained by: module galacticus_nodes

type: nodecomponentdisknull
Description: Class for the null implementation of the disk component.
Code lines: 30
Contained by: module galacticus_nodes

type: nodecomponentdiskverysimple
Description: Class for the verySimple implementation of the disk component.
Code lines: 80
Contained by: module galacticus_nodes

type: nodecomponentformationtime
Description: Type for the formationTime component class.
Code lines: 62
Contained by: module galacticus_nodes

type: nodecomponentformationtimecole2000
Description: Class for the Cole2000 implementation of the formationTime component.
Code lines: 34
Contained by: module galacticus_nodes

type: nodecomponentformationtimenull
Description: Class for the null implementation of the formationTime component.
18. Source Code Documentation

Code lines: 30
Contained by: module galacticus_nodes

type: nodecomponentthothalo
Description: Type for the hotHalo component class.
Code lines: 951
Contained by: module galacticus_nodes

type: nodecomponentthothalonull
Description: Class for the null implementation of the hotHalo component.
Code lines: 30
Contained by: module galacticus_nodes

type: nodecomponentthothalostandard
Description: Class for the standard implementation of the hotHalo component.
Code lines: 298
Contained by: module galacticus_nodes

type: nodecomponentthothaloverysimple
Description: Class for the verySimple implementation of the hotHalo component.
Code lines: 63
Contained by: module galacticus_nodes

type: nodecomponentindices
Description: Type for the indices component class.
Code lines: 69
Contained by: module galacticus_nodes

type: nodecomponentindicesnull
Description: Class for the null implementation of the indices component.
Code lines: 30
Contained by: module galacticus_nodes

type: nodecomponentindicesstandard
Description: Class for the standard implementation of the indices component.
Code lines: 49
Contained by: module galacticus_nodes

type: nodecomponentinteroutput
Description: Type for the interOutput component class.
Code lines: 139
Contained by: module galacticus_nodes

type: nodecomponentinteroutputnull
Description: Class for the null implementation of the interOutput component.
Code lines: 30
Contained by: module galacticus_nodes

type: nodecomponentinteroutputstandard
18.1. Program units

*Description:* Class for the standard implementation of the interOutput component.
*Code lines:* 72
*Contained by:* module *galacticus_nodes*

**type:** nodecomponentmergingstatistics

*Description:* Type for the mergingStatistics component class.
*Code lines:* 188
*Contained by:* module *galacticus_nodes*

**type:** nodecomponentmergingstatisticsnull

*Description:* Class for the null implementation of the mergingStatistics component.
*Code lines:* 30
*Contained by:* module *galacticus_nodes*

**type:** nodecomponentmergingstatisticsrecent

*Description:* Class for the recent implementation of the mergingStatistics component.
*Code lines:* 49
*Contained by:* module *galacticus_nodes*

**type:** nodecomponentmergingstatisticsstandard

*Description:* Class for the standard implementation of the mergingStatistics component.
*Code lines:* 100
*Contained by:* module *galacticus_nodes*

**type:** nodecomponentposition

*Description:* Type for the position component class.
*Code lines:* 125
*Contained by:* module *galacticus_nodes*

**type:** nodecomponentpositionnull

*Description:* Class for the null implementation of the position component.
*Code lines:* 30
*Contained by:* module *galacticus_nodes*

**type:** nodecomponentpositionpreset

*Description:* Class for the preset implementation of the position component.
*Code lines:* 71
*Contained by:* module *galacticus_nodes*

**type:** nodecomponentsatellite

*Description:* Type for the satellite component class.
*Code lines:* 265
*Contained by:* module *galacticus_nodes*

**type:** nodecomponentsatellitenull

*Description:* Class for the null implementation of the satellite component.
*Code lines:* 30
*Contained by:* module *galacticus_nodes*
18. Source Code Documentation

**type: nodecomponentsatellitepreset**

*Description:* Class for the preset implementation of the satellite component.

*Code lines:* 90

*Contained by:* module `galacticus_nodes`

**type: nodecomponentsatellitestandard**

*Description:* Class for the standard implementation of the satellite component.

*Code lines:* 88

*Contained by:* module `galacticus_nodes`

**type: nodecomponentsatelliteverysimple**

*Description:* Class for the verySimple implementation of the satellite component.

*Code lines:* 49

*Contained by:* module `galacticus_nodes`

**type: nodecomponentspheroid**

*Description:* Type for the `spheroid` component class.

*Code lines:* 671

*Contained by:* module `galacticus_nodes`

**type: nodecomponentspheroidnull**

*Description:* Class for the null implementation of the spheroid component.

*Code lines:* 30

*Contained by:* module `galacticus_nodes`

**type: nodecomponentspheroidstandard**

*Description:* Class for the standard implementation of the spheroid component.

*Code lines:* 279

*Contained by:* module `galacticus_nodes`

**type: nodecomponentspin**

*Description:* Type for the `spin` component class.

*Code lines:* 125

*Contained by:* module `galacticus_nodes`

**type: nodecomponentspinnull**

*Description:* Class for the null implementation of the spin component.

*Code lines:* 30

*Contained by:* module `galacticus_nodes`

**type: nodecomponentspinpreset**

*Description:* Class for the preset implementation of the spin component.

*Code lines:* 69

*Contained by:* module `galacticus_nodes`

**type: nodecomponentspinrandom**

*Description:* Class for the random implementation of the spin component.

*Code lines:* 55
18.1. Program units

**Type: nodedataabundancescalarevolvable**

*Description:* Type describing an evolvable scalar abundance property of a node component.

*Code lines:* 3

*Contained by:* module `galacticus_nodes`

**Type: nodedatachemicalabundancescalarevolvable**

*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**

*Description:* Type describing a non-evolvable scalar history property of a node component.

*Code lines:* 3

*Contained by:* module `galacticus_nodes`

**Type: nodedatahistoryscalarevolvable**

*Description:* Type describing an evolvable scalar 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: nodedatalongintegerscalar
Description: Type describing a non-evolvable scalar long integer property of a node component.
Code lines: 3
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 a 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
Contained by: module galacticus_nodes
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:* 30  
*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 typehistories.  
*Code lines:* 7  
*Contained by:* module `galacticus_nodes`  

function: **positionnullbindinginteger0in**  
*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 typehistories.  
*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 typehistories.  
*Code lines:* 7  
*Contained by:* module `galacticus_nodes`  

subroutine: **positionnullbindingsetinteger0in**  
*Description:* A null set function for rank 0 integers.
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: positionpositionhistory
Description: Returns the default value for the positionHistory property for the position component class.
Code lines: 8
Contained by: module galacticus_nodes

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: 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

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.
18.1. Program units

```plaintext
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: 16
Contained by: module galacticus_nodes

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: 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: 7
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
Modules used: galacticus_display iso_varying_string string_handling

subroutine: satellitedestroylinked
Description: Destroy the satellite component of self.
Code lines: 13
```

1085
18. Source Code Documentation

*Contained by:* module `galacticus_nodes`

**function: satelliteget**
*Description:* Returns the `satellite` component of `self`.
*Code lines:* 30
*Contained by:* module `galacticus_nodes`
*Modules used:* `galacticus_error`

**function: satellitemergetime**
*Description:* Returns the default value for the `mergeTime` property for the `satellite` component class.
*Code lines:* 20
*Contained by:* module `galacticus_nodes`

**function: satellitemergetimeisgettable**
*Description:* Returns true if the `mergeTime` property is gettable for the `satellite` component class.
*Code lines:* 9
*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: 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`

**subroutine: satellitenullbindingratedouble0inout**
*Description:* A null rate function for rank 0 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**
18.1. Program units

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.
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: satellitenullbindingsethistory0inout
Description: A null set function for rank 0 typehistorys.
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

function: satellitepresetboundmasshistoryget
Description: Return the boundMassHistory property of the SatellitePreset component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satellitepresetboundmasshistoryset
Description: Set the boundMassHistory property of the SatellitePreset component implementation.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

function: satellitepresettimeofmergingget
Description: Return the timeOfMerging property of the SatellitePreset component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: satellitepresettimeofmergingset
Description: Set the timeOfMerging property of the SatellitePreset component implementation.
Code lines: 9
18. Source Code Documentation

Contained by: module galacticus_nodes
Modules used: memory_management

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.
18.1. Program units

subroutine: satellitestandardmergetimescale
  Description: Set the mergeTime property scale of the SatelliteStandard component implementation.
  Code lines: 8
  Contained by: module galacticus_nodes

subroutine: satellitestandardmergetimeset
  Description: Set the mergeTime property of the SatelliteStandard component implementation.
  Code lines: 11
  Contained by: module galacticus_nodes
  Modules used: memory_management

function: satellitestandardvirialorbitget
  Description: Get the value of the virialOrbit property of the SatelliteStandard component using a deferred function.
  Code lines: 8
  Contained by: module galacticus_nodes

function: satellitestandardvirialorbitgetvalue
  Description: Return the virialOrbit property of the SatelliteStandard component implementation.
  Code lines: 8
  Contained by: module galacticus_nodes

subroutine: satellitestandardvirialorbitset
  Description: Set the value of the virialOrbit property of the SatelliteStandard component using a deferred function.
  Code lines: 8
  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: satellitetimeofmerging
  Description: Returns the default value for the timeOfMerging property for the satellite component class.
  Code lines: 20
  Contained by: module galacticus_nodes

function: satellitetimeofmergingisgettable
  Description: Returns true if the timeOfMerging property is gettable for the satellite component class.
  Code lines: 9
  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.
subroutine: satelliteverysimplemergetimerate
Description: Accumulate to the mergeTime property rate of change of the SatelliteVerySimple component implementation.

module: galacticus_nodes
subroutine: satelliteverysimplemergetimescale
Description: Set the mergeTime property scale of the SatelliteVerySimple component implementation.

module: galacticus_nodes
subroutine: satelliteverysimplemergetimeset
Description: Set the mergeTime property of the SatelliteVerySimple component implementation.

module: galacticus_nodes
function: satellitevirialorbit
Description: Returns the default value for the virialOrbit property for the satellite component class.

module: galacticus_nodes
subroutine: satellitevirialorbitgetfunction
Description: Set the function to be used for get of the virialOrbit property of the satellite component class.

module: galacticus_nodes
function: satellitevirialorbitgetisattached
Description: Return true if the deferred function used to get the virialOrbit property of the satellite component class has been attached.

module: galacticus_nodes
function: satellitevirialorbitisgettable
Description: Returns true if the virialOrbit property is gettable for the satellite component class.

module: galacticus_nodes
subroutine: satellitevirialorbitsetfunction
Description: Set the function to be used for set of the virialOrbit property of the satellite component class.

module: galacticus_nodes
function: satellitevirialorbitsetisattached
Description: Return true if the deferred function used to set the virialOrbit property of the satellite component class has been attached.
18.1. Program units

function: serializetoarraycount
Description: Return a count of the size of the serialized TreeNode object.
Code lines: 5
Contained by: module galacticus_nodes

subroutine: serializetoarrayrates
Description: Serialize rates to array.
Code lines: 156
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: serializetoarrayscales
Description: Serialize scales to array.
Code lines: 219
Contained by: module galacticus_nodes
Modules used: memory_management

subroutine: serializetoarrayvalues
Description: Serialize values to array.
Code lines: 219
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: 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: 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
Description: Returns the default value for the angularMomentum property for the spheroid component class.
Code lines: 8
Contained by: module galacticus_nodes

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
18.1. Program units

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
Contained by: module galacticus_nodes

function: spheroidenergygasinputisgettable
Description: Returns true if the energyGasInput property is gettable for the spheroid component class.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: spheroidenergygasinputratefunction
Description: Set the function to be used for rate of the energyGasInput property of the spheroid component class.
Code lines: 8
Contained by: module galacticus_nodes

function: spheroidenergygasinputrateisattached
Description: Return true if the deferred function used to rate the energyGasInput property of the spheroid component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes

function: spheroidget
Description: Returns the spheroid component of self.
Code lines: 30
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: 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
18. Source Code Documentation

**function:** spheroidisminitializedisgettable
- **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:** 14
- **Contained by:** module galacticus_nodes
- **Modules used:** stellar_population_properties_luminosities

**function:** spheroidluminositiesstellarisgettable
- **Description:** Returns true if the luminositiesStellar property is gettable for the spheroid component class.
- **Code lines:** 7
- **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:** spheroidmassgasisgettable
- **Description:** Returns true if the massGas property is gettable for the spheroid component class.
- **Code lines:** 6
- **Contained by:** module galacticus_nodes

**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: spheroidmassgassinksinkisgettable
Description: Returns true if the massGasSink property is gettable for the spheroid component class.
Code lines: 6
Contained by: module galacticus_nodes

subroutine: spheroidmassgassinkratefunction
Description: Set the function to be used for rate of the massGasSink property of the spheroid component class.
Code lines: 8
Contained by: module galacticus_nodes

function: spheroidmassgassinkrateisattached
Description: Return true if the deferred function used to rate the massGasSink property of the spheroid component class has been attached.
Code lines: 5
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: 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
Description: A null get function for rank 0 typeabundances.
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: spheroidnullbindingdouble1inout
Description: A null get function for rank 1 double precisions.
Code lines: 7
Contained by: module galacticus_nodes
function: spheroidnullbindinghistory0inout
Description: A null get function for rank 0 type historys.
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: spheroidnullbindingratedoublenew
Description: A null rate function for rank 1 double precisions.
Code lines: 9
Contained by: module galacticus_nodes

subroutine: spheroidnullbindingratehistory0inout
Description: A null rate function for rank 0 type historys.
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

subroutine: spheroidnullbindingsetabundances0inout
Description: A null set function for rank 0 type abundances.
Code lines: 7
18.1. Program units

**module** galacticus_nodes

**subroutine** spheroidnullbindingsetdouble0inout
*Description:* A null set function for rank 0 double precisions.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine** spheroidnullbindingsetdouble1inout
*Description:* A null set function for rank 1 double precisions.
*Code lines:* 7
*Contained by:* module galacticus_nodes

**subroutine** spheroidnullbindingsethistory0inout
*Description:* A null set function for rank 0 typehistorys.
*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

**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** 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
18. Source Code Documentation

**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
18.1. Program units

Description: Set the \texttt{abundancesStellar} property scale of the \texttt{SpheroidStandard} component implementation.

\begin{verbatim}
Code lines: 8
Contained by: module \texttt{galacticus_nodes}
\end{verbatim}

\textbf{subroutine: spheroidstandardabundancesstellarset}
\begin{verbatim}
Description: Set the \texttt{abundancesStellar} property of the \texttt{SpheroidStandard} component implementation.
Code lines: 11
Contained by: module \texttt{galacticus_nodes}
\end{verbatim}

\textbf{Modules used: memory_management}

\textbf{function: spheroidstandardangularmomentumcount}
\begin{verbatim}
Description: Return a count of the number of scalar properties in the \texttt{angularMomentum} property of the \texttt{spheroidStandard} component implementation.
Code lines: 7
Contained by: module \texttt{galacticus_nodes}
\end{verbatim}

\textbf{function: spheroidstandardangularmomentumget}
\begin{verbatim}
Description: Return the \texttt{angularMomentum} property of the \texttt{SpheroidStandard} component implementation.
Code lines: 8
Contained by: module \texttt{galacticus_nodes}
\end{verbatim}

\textbf{subroutine: spheroidstandardangularmomentumrate}
\begin{verbatim}
Description: Accumulate to the \texttt{angularMomentum} property rate of change of the \texttt{SpheroidStandard} component implementation.
Code lines: 10
Contained by: module \texttt{galacticus_nodes}
\end{verbatim}

\textbf{subroutine: spheroidstandardangularmomentumrategeneric}
\begin{verbatim}
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}
\end{verbatim}

\textbf{subroutine: spheroidstandardangularmomentumscale}
\begin{verbatim}
Description: Set the \texttt{angularMomentum} property scale of the \texttt{SpheroidStandard} component implementation.
Code lines: 8
Contained by: module \texttt{galacticus_nodes}
\end{verbatim}

\textbf{subroutine: spheroidstandardangularmomentumset}
\begin{verbatim}
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}
\end{verbatim}

\textbf{subroutine: spheroidstandardcreatefunctionset}
\begin{verbatim}
Description: Set the create function for the \texttt{SpheroidStandard} component.
\end{verbatim}
18. Source Code Documentation

subroutine: spheroidstandardenergygasinputrate
Description: Set the rate of the energyGasInput property of the SpheroidStandard component using a deferred function.
Code lines: 10
Contained by: module galacticus_nodes

function: spheroidstandardisinitializedget
Description: Return the isInitialized property of the SpheroidStandard component implementation.
Code lines: 8
Contained by: module galacticus_nodes

subroutine: spheroidstandardisinitializedset
Description: Set the isInitialized property of the SpheroidStandard component implementation.
Code lines: 9
Contained by: module galacticus_nodes
Modules used: memory_management

function: spheroidstandardluminositiesstellarcount
Description: Return a count of the number of scalar properties in the luminositiesStellar property of the spheroidStandard component implementation.
Code lines: 11
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: 35
   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

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

function: spheroidstandardmassstellarcount
18. Source Code Documentation

**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: spheroidstandardmassstellelarget**

**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

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

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

function: spheroidstarformationhistoryisgettable
Description: Returns true if the starFormationHistory property is gettable for the spheroid component class.
Code lines: 6
Contained by: module galacticus_nodes
18.1. Program units

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

subroutine: spheroidstarformationrategetfunction
Description: Set the function to be used for get of the starFormationRate property of the spheroid component class.
Code lines: 8
Contained by: module galacticus_nodes

function: spheroidstarformationrategetisattached
Description: Return true if the deferred function used to get the starFormationRate property of the spheroid component class has been attached.
Code lines: 5
Contained by: module galacticus_nodes

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: 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: 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: 30  
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: spinnullbindinginteger0in  
Description: A null get function for rank 0 integers.  
Code lines: 7  
Contained by: module galacticus_nodes
18.1. Program units

subroutine: spinnullbindingratedouble0inout
  Description: A null rate function for rank 0 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: spinnullbindingsetinteger0in
  Description: A null set function for rank 0 integers.
  Code lines: 7
  Contained by: module galacticus_nodes

function: spinpresetspincount
  Description: Return a count of the number of scalar properties in the spin property of the spinPreset component implementation.
  Code lines: 7
  Contained by: module galacticus_nodes

function: spinpresetspinget
  Description: Return the spin property of the SpinPreset component implementation.
  Code lines: 8
  Contained by: module galacticus_nodes

function: spinpresetspingrowthrateget
  Description: Return the spinGrowthRate property of the SpinPreset component implementation.
  Code lines: 8
  Contained by: module galacticus_nodes

subroutine: spinpresetspingrowthrateset
  Description: Set the spinGrowthRate property of the SpinPreset component implementation.
  Code lines: 9
  Contained by: module galacticus_nodes
  Modules used: memory_management

subroutine: spinpresetspinrate
  Description: Accumulate to the spin property rate of change of the SpinPreset component implementation.
  Code lines: 10
  Contained by: module galacticus_nodes
18. Source Code Documentation

**subroutine: spinpresetspin**

*Description:* Set the *spin* property scale of the *SpinPreset* component implementation.

*Code lines:* 8

*Contained by:* module galacticus_nodes

**function: spinrandomspin**

*Description:* Return the default value for the *spin* property for the *spin* component class.

*Code lines:* 8

*Contained by:* module galacticus_nodes

**module:** galacticus_nodes

**function: spin**

*Description:* Returns the default value for the *spin* property for the *spin* component class.

*Code lines:* 8

*Contained by:* module galacticus_nodes

**function: spinGrowthRate**

*Description:* Returns the default value for the *spinGrowthRate* property for the *spin* component class.

*Code lines:* 8

*Contained by:* module galacticus_nodes

**function: spinrandomspin**

*Description:* Return a count of the number of scalar properties in the *spin* property of the *spinRandom* component implementation.

*Code lines:* 7

*Contained by:* module galacticus_nodes

**function: spinrandomspin**

*Description:* Return the *spin* property of the *SpinRandom* component implementation.

*Code lines:* 8

*Contained by:* module galacticus_nodes

**function: spinrandomspin**

*Description:* Return the *spin* property of the *SpinRandom* component implementation.

*Code lines:* 8

*Contained by:* module galacticus_nodes

**function: spinrandomspin**

*Description:* Accumulate to the *spin* property rate of change of the *SpinRandom* component implementation.

*Code lines:* 10

*Contained by:* module galacticus_nodes

**function: spinrandomspin**

*Description:* Set the *spin* property of the *SpinRandom* component implementation.

*Code lines:* 8

*Contained by:* module galacticus_nodes

**function: spinrandomspin**

*Description:* Set the *spin* property scale of the *SpinRandom* component implementation.

*Code lines:* 8

*Contained by:* module galacticus_nodes

**function: spinrandomspin**

*Description:* Set the *spin* property of the *SpinRandom* component implementation.

*Code lines:* 11

*Contained by:* module galacticus_nodes

**function: spinrandomspin**

*Description:* Set the *spin* property scale of the *SpinRandom* component implementation.

*Code lines:* 8

*Contained by:* module galacticus_nodes

**function: spinrandomspin**

*Description:* Set the *spin* property of the *SpinRandom* component implementation.

*Code lines:* 11

*Modules used:* memory_management

**function: spin**

*Description:* Returns the default value for the *spin* property for the *spin* component class.

*Code lines:* 8

*Contained by:* module galacticus_nodes

**function: spinGrowthRate**

*Description:* Returns the default value for the *spinGrowthRate* property for the *spin* component class.

*Code lines:* 8

*Contained by:* module galacticus_nodes
18.1. Program units

function: spinGrowthRateIsGettable

Description: Returns true if the spinGrowthRate property is gettable for the spin component class.

Code lines: 6

Contained by: module galacticus_nodes

function: spinIsGettable

Description: Returns true if the spin property is gettable for the spin component class.

Code lines: 6

Contained by: module galacticus_nodes

subroutine: treeNodeComponentBuilder

Description: Build components in a treeNode object given an XML definition.

Code lines: 261

Contained by: module galacticus_nodes

Modules used: fox_dom

function: treeNodeConstructor

Description: Return a pointer to a newly created and initialized treeNode.

Code lines: 19

Contained by: module galacticus_nodes

subroutine: treeNodeCopyNodeTo

Description: Make a copy of self in targetNode.

Code lines: 362

Contained by: module galacticus_nodes

function: treeNodeCreateEvent

Description: Create a new event in a tree node.

Code lines: 21

Contained by: module galacticus_nodes

function: treeNodeGetEarliestProgenitor

Description: Returns a pointer to the earliest progenitor of self.

Code lines: 14

Contained by: module galacticus_nodes

function: treeNodeGetLastSatellite

Description: Returns a pointer to the final satellite node associated with self.

Code lines: 11

Contained by: module galacticus_nodes

function: treeNodeIndex

Description: Returns the index of a treeNode.

Code lines: 17

Contained by: module galacticus_nodes

subroutine: treeNodeIndexSet

Description: Sets the index of a treeNode.
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_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: 164
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: 155
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: 155
Contained by: module galacticus_nodes

subroutine: tree_node_remove_from_host
18.1. Program units

**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`
18.1. Program units

**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:** `treenode`
**Description:** A class for `node` in merger trees.
**Code lines:** 881
**Contained by:** module `galacticus_nodes`

**subroutine:** `treenode_destroy`
**Description:** Destroy a `treenode` object.
**Code lines:** 52
**Contained by:** module `galacticus_nodes`

**subroutine:** `treenode_initialize`
**Description:** Initialize a `treenode` object.
**Code lines:** 58
**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`

**file:** `objects.nodes.bindings.C.F90`
**Code lines:** 718

**function:** `cnode_component_basic_accretionrate`
**Code lines:** 13
**Contained by:** file `objects.nodes.bindings.C.F90`
Modules used: galacticus_nodes isoc_binding

function: cnode_component_basic_mass
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes isoc Binding

function: cnode_component_basic_time
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes isoc Binding

function: cnode_component_basic_timelastisolated
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes isoc Binding

function: cnode_component_blackhole_mass
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes isoc Binding

function: cnode_component_blackhole_massseed
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes isoc Binding

function: cnode_component_blackhole_radialposition
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes isoc Binding

function: cnode_component_blackhole_spin
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes isoc Binding

function: cnode_component_blackhole_spindseed
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes isoc Binding

function: cnode_component_blackhole_tripleinteractiontime
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes isoc Binding

function: cnode_component_darkmatterprofile_scale
Code lines: 13
18.1. Program units

```
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
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
```
### 18. Source Code Documentation

| Code lines | 13 |
| Contains by | file `objects.nodes.bindings.C.F90` |
| Modules used | `galacticus_nodes` iso_c_binding |

**function: cnode_component_formationtime_formationtime**

| Code lines | 13 |
| Contains by | file `objects.nodes.bindings.C.F90` |
| Modules used | `galacticus_nodes` iso_c_binding |

**function: cnode_component_hothalo_angularmomentum**

| Code lines | 13 |
| Contains by | file `objects.nodes.bindings.C.F90` |
| Modules used | `galacticus_nodes` iso_c_binding |

**function: cnode_component_hothalo_mass**

| Code lines | 13 |
| Contains by | file `objects.nodes.bindings.C.F90` |
| Modules used | `galacticus_nodes` iso_c_binding |

**function: cnode_component_hothalo_outerradius**

| Code lines | 13 |
| Contains by | file `objects.nodes.bindings.C.F90` |
| Modules used | `galacticus_nodes` iso_c_binding |

**function: cnode_component_hothalo_outflowedangularmomentum**

| Code lines | 13 |
| Contains by | file `objects.nodes.bindings.C.F90` |
| Modules used | `galacticus_nodes` iso_c_binding |

**function: cnode_component_hothalo_outflowedmass**

| Code lines | 13 |
| Contains by | file `objects.nodes.bindings.C.F90` |
| Modules used | `galacticus_nodes` iso_c_binding |

**function: cnode_component_hothalo_strippedmass**

| Code lines | 13 |
| Contains by | file `objects.nodes.bindings.C.F90` |
| Modules used | `galacticus_nodes` iso_c_binding |

**function: cnode_component_hothalo_unaccretedmass**

| Code lines | 13 |
| Contains by | file `objects.nodes.bindings.C.F90` |
| Modules used | `galacticus_nodes` iso_c_binding |

**function: cnode_component_interoutput_diskstarformationrate**

| Code lines | 13 |
| Contains 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_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_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_timeofmerging
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes iso_c_binding

function: cnode_component_spheroid_angularmomentum
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes iso_c_binding

function: cnode_component_spheroid_halfmassradius
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
Modules used: galacticus_nodes iso_c_binding

function: cnode_component_spheroid_massgas
Code lines: 13
Contained by: file objects.nodes.bindings.C.F90
18. Source Code Documentation

**Modules used:** galacticus_nodes iso_c_binding

**function:** cnode_component_spheroid_massstellar
  **Code lines:** 13
  **Contained by:** file objects.nodes.bindings.C.F90
  **Modules used:** galacticus_nodes iso_c_binding

**function:** cnode_component_spheroid_radius
  **Code lines:** 13
  **Contained by:** file objects.nodes.bindings.C.F90
  **Modules used:** galacticus_nodes iso_c_binding

**function:** cnode_component_spheroid_starformationrate
  **Code lines:** 13
  **Contained by:** file objects.nodes.bindings.C.F90
  **Modules used:** galacticus_nodes iso_c_binding

**function:** cnode_component_spheroid_velocity
  **Code lines:** 13
  **Contained by:** file objects.nodes.bindings.C.F90
  **Modules used:** galacticus_nodes iso_c_binding

**function:** cnode_component_spin_spin
  **Code lines:** 13
  **Contained by:** file objects.nodes.bindings.C.F90
  **Modules used:** galacticus_nodes iso_c_binding

**function:** cnode_component_spingrowthrate
  **Code lines:** 13
  **Contained by:** file objects.nodes.bindings.C.F90
  **Modules used:** galacticus_nodes iso_c_binding

**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
18.1. Program units

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
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
18. Source Code Documentation

**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.black_hole.simple.F90
- **Description:** Contains a module which implements the simple black hole node component.
- **Code lines:** 434

**module:** node_component_black_hole_simple
- **Description:** Implements the simple black hole node component.
- **Code lines:** 414
- **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
18.1. Program units

**subroutine: node_component_black_hole_simple_output_count**
*Description:* Account for the number of black hole properties to be written to the 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.
*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:* 74
*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 subroutine galactic_structure_rotation_curve_gradient

**function: node_component_black_hole_simple_potential**
*Description:* Compute the gravitational potential due to a black hole.
*Code lines:* 28
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

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

file: objects.nodes.components.black_hole.standard.F90
Description: Contains a module which implements the standard black hole node component.
Code lines: 1114

module: node_component_black_hole_standard
Description: Implement black hole tree node methods.
Code lines: 1094

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: 20

subroutine: node_component_black_hole_standard_create
Description: Creates a black hole component for thisNode.
Code lines: 13
subroutine: node_component_black_hole_standard_initialize
Description: Initializes the standard black hole component module.
Code lines: 159
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: 104
Contained by: module node_component_black_hole_standard
Modules used: accretion_disks black_hole_fundamentals galactic_structure_densities hot_halo_temperature_profile 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

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 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
18. Source Code Documentation

**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
**Description:** Output properties for all black holes in thisNode.
**Code lines:** 88
**Contained by:** module node_component_black_hole_standard
**Modules used:** accretion_disks, black_hole_binary_separations, galacticus_hdf5, iso_varying_string, kind_numbers, memory_management, string_handling

**subroutine:** node_component_black_hole_standard_rate_compute
**Description:** Compute the black hole node mass rate of change.
**Code lines:** 167
**Contained by:** module node_component_black_hole_standard
**Modules used:** accretion_disks, black_hole_binary_separations, cosmological_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
**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
18.1. Program units

**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_solve_initial_adiabatic`
- 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:** 333

**module: node_component_dark_matter_profile_scale**

**Description:** Implements a dark matter profile method that provides a scale radius.

**Code lines:** 313

**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 `galactic_structure_radii_solve_linear`
- subroutine `get_tree`
- subroutine `tree_node_compute_derivatives`
subroutine tree_node_evolve subroutine tree_node_promote
subroutine merger_tree_initialize subroutine merger_tree_structure_output

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: 21
Contained by: module node_component_dark_matter_profile_scale
Modules used: dark_matter_halo_scales dark_matter_profiles_concentrations

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: 24
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: 13
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
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_shape.F90
Description: Contains a module which implements a dark matter profile method that provides a scale radius and a shape parameter.
Code lines: 250

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_evolve subroutine tree_node_promote subroutine merger_tree_initialize subroutine merger_tree_structure_output

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
18. Source Code Documentation

**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:** 1045

**module:** `node_component_disk_exponential`
- **Description:** Implements the exponential disk node component.
- **Code lines:** 1025
- **Contained by:** file `objects.nodes.components.disk.exponential.F90`
- **Modules used:** `galacticus_nodes`
- **Used by:**
  - subroutine `galacticus_structure_radii._solve_adiabatic`
  - subroutine `galacticus_structure_radii._solve_linear`
  - subroutine `galacticus_calculations._reset`
  - subroutine `get_tree`
  - subroutine `tree_node_evolve`
  - subroutine `node_component_disk_exponential_create`
  - subroutine `galacticus_output_star_formation._histories`
  - subroutine `stellar_population_properties`

**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`
  - subroutine `stellar_population_properties`

**subroutine:** `node_component_disk_exponential_initialize`
- **Description:** Initializes the tree node exponential disk methods module.
- **Code lines:** 119
18.1. Program units

```
Contained by:  module node_component_disk_exponential
Modules used:  abundances_structure  input_parameters
               memory_management   node_component_disk_exponential_data
               stellar_population_properties_luminosities

subroutine: node_component_disk_exponential_post_evolve
Description:  Trim histories attached to the disk.
Code lines:  101
Contained by:  module node_component_disk_exponential
Modules used:  abundances_structure  dark_matter_halo_scales
               galacticus_display  galacticus_error
               histories  iso_varying_string
               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:  37
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

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:  37
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:  184
Contained by:  module node_component_disk_exponential
Modules used:  abundances_structure  galactic_dynamics_bar_instabilities
               galactic_structure_options  galacticus_output_star_formation_histories
```
18. Source Code Documentation

histories numerical_constants_astronomical
ram_pressure_stripping_mass_loss_rate_disks
star_formation_feedback_expulsion_disks
stellar_population_properties

subroutine: node_component_disk_exponential_satellite_merging
Description: Transfer any exponential disk associated with thisNode to its host halo.
Code lines: 99
Contained by: module node_component_disk_exponential
Modules used: abundances_structure galacticus_error
histories satellite_merging_mass_movements_descriptors

subroutine: node_component_disk_exponential_scale_set
Description: Set scales for properties of thisNode.
Code lines: 60
Contained by: module node_component_disk_exponential
Modules used: abundances_structure galacticus_output_star_formation_histories
histories stellar_population_properties

subroutine: node_component_disk_exponential_star_formation_history_output
Description: Store the star formation history in the output file.
Code lines: 22
Contained by: module node_component_disk_exponential
Modules used: galacticus_output_star_formation_histories histories 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: 222
module: node_component_disk_exponential_data
Code lines: 202
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 subroutine node_component_disk_- exponential_calculation_reset subroutine node_component_disk_- exponential_initialize 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
Modules used: fgsl

file: objects.nodes.components.disk.very_simple.F90
Description: Contains a module that implements a very simple disk component.
module: node_component_disk_very_simple
Description: Implements a very simple disk component.
Code lines: 299
Contained by: file objects.nodes.components.disk.very_simple.F90
Modules used: galacticus_nodes iso_varying_string
Used by: subroutine get_tree subroutine tree_node_compute_derivatives
subroutine tree_node_evolve subroutine satellite_merger_process

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: 47
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: 29
Contained by: module node_component_disk_very_simple
Modules used: dark_matter_halo_scales star_formation_timescales_disks
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
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.hot_halo.standard.F90
Description: Contains a module which implements the standard hot halo node component.
Code lines: 1296

module: node_component_hot_halo_standard
Description: Implements the standard hot halo node component.
Code lines: 1276
18. Source Code Documentation

**Contained by:** file `objects.nodes.components.hot_halo.standard.F90`

**Modules used:**
- `galacticus_nodes`
- `galacticus_error`
- `galacticus_calculations`
- `galacticus_calculations.base`
- `radiation_structure`

**Used by:**
- subroutine `event_halo_formation`
- subroutine `galacticus_calculations.reset`
- subroutine `get_tree`
- subroutine `events_node_merger`
- subroutine `tree_node_evolve`
- subroutine `tree_node_evolve`
- subroutine `merger_tree_evolve_to`
- subroutine `merger_tree_evolve`
- subroutine `merger_tree_evolve`
- subroutine `merger_tree_evolve`
- subroutine `tree_node_evolve`
- subroutine `merger_tree_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:** 12

**Contained by:** module `node_component_hot_halo_standard`

**Modules used:**
- `dark_matter_halo_scales`

**subroutine: node_component_hot_halo_standard_formation**

**Description:** Updates the hot halo gas distribution at a formation event, if requested.

**Code lines:** 56

**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`
- `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:** 46

**Contained by:** module `node_component_hot_halo_standard`

**Modules used:**
- `dark_matter_halo_scales`
- `galacticus_error`

**subroutine: node_component_hot_halo_standard_initialize**

**Description:** Initializes the standard hot halo component module.

**Code lines:** 192

**Contained by:** module `node_component_hot_halo_standard`

**Modules used:**
- `abundances_structure`
- `galacticus_error`
- `input_parameters`
- `iso_varying_string`

**subroutine: node_component_hot_halo_standard_node_merger**

**Description:** Starve `thisNode` by transferring its hot halo to its parent.

**Code lines:** 68

**Contained by:** module `node_component_hot_halo_standard`
18.1. Program units

Modules used:
- abundances_structure
- chemical_abundances_structure
- cosmological_parameters
- dark_matter_halo_scales
- galactic_structure_enclosed_masses
- galactic_structure_options

function: node_component_hot_halo_standard_outer_radius
Description: Return the outer radius in the standard hot halo.
Code lines: 10
Contained by: module node_component_hot_halo_standard
Modules used: dark_matter_halo_scales

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: 20
Contained by: module node_component_hot_halo_standard
Modules used: dark_matter_halo_scales
- hot_halo_density_profile

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: 22
Contained by: module node_component_hot_halo_standard

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: 52
Contained by: module node_component_hot_halo_standard
Modules used: abundances_structure
- dark_matter_halo_scales

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: 34
Contained by: module node_component_hot_halo_standard
Modules used: 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.
subroutine: node_component_hot_halo_standard_push_to_null
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.

subroutine: node_component_hot_halo_standard_rate_compute
Description: Compute the hot halo node mass rate of change.

subroutine: node_component_hot_halo_standard_reset
Description: Remove memory of stored computed values as we’re about to begin computing derivatives anew.

subroutine: node_component_hot_halo_standard_satellite_merger
Description: Remove any hot halo associated with thisNode before it merges with its host halo.

subroutine: node_component_hot_halo_standard_scale_set
Description: Set scales for properties of thisNode.

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).
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
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: 46
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.very_simple.F90
Description: Contains a module which implements a very simple hot halo node component.
Code lines: 378

module: node_component_hot_halo_very_simple
Description: Implements a very simple hot halo node component.
Code lines: 358
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_promote subroutine tree_node_evolve 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: 18
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
18. Source Code Documentation

```
Contained by: module node_component_hot_halo_very_simple

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

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: 40
Contained by: module node_component_hot_halo_very_simple
Modules used: cosmological_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: 167

module: node_component_inter_output_standard
Description: Implements the standard indices component.
Code lines: 147
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 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: 18
Contained by: module node_component_inter_output_standard
Modules used: 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
file: `objects.nodes.components.merging_statistics.recent.F90`

*Description:* Contains a module which implements the recent merging statistics component.

*Code lines:* 340

**module:** `node_component_merging_statistics_recent`

*Description:* Implements the recent merging statistics component.

*Code lines:* 320

*Contained by:* file `objects.nodes.components.merging_statistics.recent.F90`

*Modules used:* `galacticus_nodes`

*Used by:* subroutine `count_properties` subroutine `establish_property_names` subroutine `galacticus_merger_tree_-output` subroutine `events_node_merger` 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` `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:* 59

*Contained by:* module `node_component_merging_statistics_recent`

*Modules used:* `dark_matter_halo_scales` `galacticus_error` `galacticus_output_times`

**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.
18.1. Program units

**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:** 22
- **Modules used:** galacticus_output_times kind_numbers
- **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:** 9
- **Modules used:** galacticus_output_times kind_numbers
- **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:** 216

**module:** node_component_merging_statistics_standard
- **Description:** Implements the standard merging statistics component.
- **Code lines:** 196
- **Modules used:** galacticus_nodes
- **Used by:** subroutine events_node_merger subroutine tree_node_promote subroutine satellite_merger_process subroutine merger_tree_initialize
- **Contained by:** file objects.nodes.components.merging_statistics.standard.F90

**subroutine:** node_component_merging_statistics_standard_initialize
- **Description:** Initializes the standard merging statistics component.
- **Code lines:** 35
- **Modules used:** input_parameters
- **Contained by:** module node_component_merging_statistics_standard

**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:** 31
- **Modules used:** dark_matter_halo_formation_times
- **Contained by:** module node_component_merging_statistics_standard

**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.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.preset.F90
Description: Contains a module which implements a preset satellite orbit component.
Code lines: 106

module: node_component_satellite_preset
Description: Implements a preset satellite orbit component.
Code lines: 84
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.
Code lines: 22
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: 328

module: node_component_satellite_standard
Description: Implements the standard satellite component.
Code lines: 309
18.1. Program units

Contained by: file objects.nodes.components.satellite.standard.F90
Modules used: galacticus_nodes kepler_orbits

Used by: subroutine event_halo_formation subroutine get_tree
           subroutine events_node_merger subroutine tree_node_compute_derivatives
           subroutine tree_node_evolve subroutine merger_tree_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: 50
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.
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
18. Source Code Documentation

**subroutine**: node_component_satellite_standard_virial_orbit_set

*Description*: Set the orbit of the satellite at the virial radius.

*Code lines*: 24

*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*: 177

**module**: node_component_satellite_very_simple

*Description*: Implements a very simple satellite orbit component.

*Code lines*: 157

*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*: 40

*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

*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*: 1120

**module**: node_component_spheroid_standard

*Description*: Implements the standard spheroid component.
18.1. Program units

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 $M_\odot \cdot km^2 \cdot s^{-2} \cdot Gyr^{-1}$.
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: 139
Contained by: module node_component_spheroid_standard
Modules used: abundances_structure galacticus_error

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

subroutine: node_component_spheroid_standard_post_evolve
Description: Trim histories attached to the spheroid.
Code lines: 76
Contained by: module node_component_spheroid_standard
Modules used: abundances_structure galacticus_display
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: 92
Contained by: module node_component_spheroid_standard
Modules used: abundances_structure galactic_structure_options
galacticus_output_star_formation_histories
star_formation_feedback_expulsion_spheroids
star_formation_feedback_spheroids
numerical_constants_astronomical

subroutine: node_component_spheroid_standard_satellite_merging
Description: Transfer any standard spheroid associated with thisNode to its host halo.
Code lines: 209
Contained by: module node_component_spheroid_standard
Modules used: abundances_structure galacticus_error
satellite_merging_mass_movements_descriptors
satellite_merging_remnant_sizes_properties

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: 59
18.1. Program units

**Contained by:** module node_component_spheroid_standard

**Modules used:**
- abundances_structure
- galacticus_output_star_formation_histories

**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

**Contained by:** module 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:** 21

**Contained by:** module node_component_spheroid_standard

**Modules used:**
- galacticus_output_star_formation_histories
- kind_numbers

**Subroutine:** node_component_spheroid_standard_star_formation_history_rate

**Description:** Adjust the rates for the star formation history.

**Code lines:** 27

**Contained by:** module node_component_spheroid_standard

**Function:** node_component_spheroid_standard_star_formation_rate

**Description:** Return the star formation rate of the standard spheroid.

**Code lines:** 24

**Contained by:** module node_component_spheroid_standard

**Modules used:**
- star_formation_timescales_spheroids

**Function:** node_component_spheroid_standard_velocity_solve

**Description:** Return the circular velocity of the standard spheroid.

**Code lines:** 9

**Contained by:** module 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:** 32

**Module:** node_component_spheroid_standard_data

**Description:** Contains data for standard spheroid components.

**Code lines:** 12

**Contained by:** file objects.nodes.components.spheroid.standard.data.F90

**Modules used:**
- mass_distributions

**Used by:** 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_evolve subroutine tree_node_promote 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.
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.random.F90
Description: Contains a module of spin tree node methods.
Code lines: 165

module: node_component_spin_random
Description: Implement random spin tree node method.
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.
18.1. Program units

**Code lines:** 25  
**Contained by:** module node_component_spin_random  
**Modules used:** input_parameters

**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:** 347  
**Contained by:** file objects.radiation.F90  
**Used by:** module accretion_halos_simple  
**subroutine:** chemical_hydrogen_rate_h2_-electron_to_2h_electron  
**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_electron_to_2h  
**subroutine:** chemical_hydrogen_rate_h2plus_gamma_to_2hplus_electron  
**subroutine:** chemical_hydrogen_rate_h2plus_h_to_2hplus  
**subroutine:** chemical_hydrogen_rate_h2plus_hminus_to_h2h  
**subroutine:** chemical_hydrogen_rate_h2plus_electron_to_hplus  
**subroutine:** chemical_hydrogen_rate_h2plus_hminus_to_h2h  
**subroutine:** chemical_hydrogen_rate_h2plus_hminus_to_h2h  
**subroutine:** chemical_hydrogen_rate_h2plus_hminus_to_h2h  
**subroutine:** chemical_hydrogen_rate_h2plus_hminus_to_h2h  
**subroutine:** chemical_hydrogen_rate_h2plus_hminus_to_h2h  
**subroutine:** chemical_hydrogen_rate_h2plus_hminus_to_h2h

**module:** radiation_structure  
**Description:** Defines the radiation structure data type, used to describe radiation fields. (Currently only includes CMB radiation temperature.)  
**Code lines:** 326  
**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_electron_to_2h  
**subroutine:** chemical_hydrogen_rate_h2plus_gamma_to_2hplus_electron  
**subroutine:** chemical_hydrogen_rate_h2plus_h_to_2hplus  
**subroutine:** chemical_hydrogen_rate_hminus_electron_to_h2h_electron  
**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_2h  
**subroutine:** chemical_hydrogen_rate_hminus_hplus_to_2h  
**subroutine:** chemical_hydrogen_rate_hminus_hplus_to_2h  
**subroutine:** chemical_hydrogen_rate_hminus_hplus_to_2h
subroutine chemical_hydrogen_rate_hminus_hplus_to_h2plus_electron
subroutine chemical_hydrogen_rates_compute
subroutine chemical_densities_cie_file
function electron_density_cie_file
function electron_density_cie_file_logtemperature_interpolate
function electron_density_temperature_log_slope_cie_file
subroutine chemical_densities_atomic_cie_cloudy
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_cloudy
subroutine cooling_function_atomic_cie_cloudy
subroutine cooling_function_density_slope_atomic_cie_cloudy
module chemical_states

function cooling_function_cmb_compton
subroutine cooling_function_density_slope_molecular_hydrogen_gp
subroutine cooling_function_molecular_hydrogen_gp
subroutine cooling_function_temperature_slope_molecular_hydrogen_gp
module cooling_radii_isothermal

module cooling_radii_simple

subroutine cooling_function_atomic_cie_cloudy
subroutine cooling_function_temperature_slope_atomic_cie_cloudy
subroutine cooling_function_atomic_cie_cloudy
subroutine cooling_function_density_slope_molecular_hydrogen_gp
subroutine cooling_function_molecular_hydrogen_gp
subroutine cooling_function_temperature_slope_molecular_hydrogen_gp
module cooling_radii_isothermal

function cooling_time_density_log_slope_simple
function cooling_time_temperature_log_slope_simple
module node_component_hot_halo_standard

function: cross_section_integrand
Description: Integrand function use 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
18.1. Program units

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$^2$ s$^{-1}$ 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.:

$$\frac{4\pi}{\lambda_2^2} \int_{\lambda_1}^{\lambda_2} \sigma(\lambda) j_\nu(\lambda) \frac{d\lambda}{\lambda},$$ (18.9)

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
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

function: radiation_temperature
Description: Return the temperature of the radiation object.
Code lines: 40
Contained by: module radiation_structure
Modules used: radiation_cmb radiation_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
18. Source Code Documentation

**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:** 62

**Contained by:** module `radiation_structure`

**file: objects.tables.F90**

**Description:** Contains a module which defines a table class with optimized interpolation operators.

**Code lines:** 1008

**module: tables**

**Description:** Defines a table class with optimized interpolation operators.

**Code lines:** 988

**Contained by:** file `objects.tables.F90`

**Modules used:** fgs1

**Used by:**

- module `accretion_disks_adaf`
- module `halo_spin_distributions_bet2007`
- module `conditional_stellar_mass_functions_behroozi2010`
- module `intergalactic_medium_state`
- module `virial_orbits_wetzel2010`
- subroutine `star_formation_imf_tabulate_chabrier`
- subroutine `star_formation_imf_tabulate_kennicutt`
- subroutine `star_formation_imf_tabulate_millerscalo`
- subroutine `star_formation_imf_tabulate_salpeter`
- subroutine `star_formation_imf_tabulate_scalo`
- subroutine `star_formation_imf_tabulate_piecewisepowerlaw`
- subroutine `stellar_population_spectra_conroy_initialize_imf`
- module `critical_overdensity`
- module `power_spectra`
- subroutine `sphericalCollapseDark_energy_critical_overdensity`
- subroutine `make_table`
- subroutine `sphericalCollapseVirial_density_contrast`
module virial_density_contrast subroutine virial_density_kitayama_suto1996 subroutine virial_density_fixed program test_tables

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
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 \( i^{\text{th}} \) \( x \)-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 \( i^{\text{th}} \) \( y \)-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
18. Source Code Documentation

**Contained by:** module tables

**type:** table1dgeneric  
*Description:* Table type supporting generic one dimensional tables.  
*Code lines:* 28  
*Contained by:* module tables

**type:** table1dlinearxcspline  
*Description:* Table type supporting one dimensional table with linear spacing in \( x \) and cubic spline interpolation.  
*Code lines:* 29  
*Contained by:* module tables

**type:** table1dlinearxlinear  
*Description:* Table type supporting one dimensional table with linear spacing in \( x \).  
*Code lines:* 26  
*Contained by:* module tables

**type:** table1dlogarithmicxcspline  
*Description:* Table type supporting one dimensional table with logarithmic spacing in \( x \) and cubic spline interpolation.  
*Code lines:* 8  
*Contained by:* module tables

**type:** table1dlogarithmicxlinear  
*Description:* Table type supporting one dimensional table with logarithmic spacing in \( x \).  
*Code lines:* 8  
*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

1154
18.1. Program units

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`

**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: 27
Contained by: module `tables`
Modules used: `memory_management` `numerical_ranges`

**function: table_linear_1d_interpolate**
Description: Perform linear interpolation in a linear 1D table.
Code lines: 32
Contained by: module `tables`

**function: table_linear_1d_interpolate_gradient**
Description: Perform linear interpolation in a linear 1D table.
Code lines: 29
Contained by: module `tables`

**subroutine: table_linear_1d_populate**
Description: Populate a 1-D linear table.
Code lines: 24
Contained by: module `tables`
18. Source Code Documentation

**Modules used:** \texttt{galacticus_error}

**subroutine:** \texttt{table\_linear\_1d\_populate\_single}
\textbf{Description:} Populate a single element of a 1-D linear table.
\textbf{Code lines:} 25
\textbf{Contained by:} module \texttt{tables}
\textbf{Modules used:} \texttt{galacticus_error}

**subroutine:** \texttt{table\_linear\_cspline\_1d\_coefficients}
\textbf{Description:} Compute coefficients for a spline interpolation.
\textbf{Code lines:} 21
\textbf{Contained by:} module \texttt{tables}

**subroutine:** \texttt{table\_linear\_cspline\_1d\_compute\_spline}
\textbf{Description:} Compute the interpolating spline factors for a 1-D linear spline.
\textbf{Code lines:} 34
\textbf{Contained by:} module \texttt{tables}

**subroutine:** \texttt{table\_linear\_cspline\_1d\_create}
\textbf{Description:} Create a 1-D linear table.
\textbf{Code lines:} 29
\textbf{Contained by:} module \texttt{tables}
\textbf{Modules used:} \texttt{memory\_management numerical\_ranges}

**subroutine:** \texttt{table\_linear\_cspline\_1d\_destroy}
\textbf{Description:} Destroy a linear cubic-spline 1-D table.
\textbf{Code lines:} 9
\textbf{Contained by:} module \texttt{tables}
\textbf{Modules used:} \texttt{memory\_management}

**function:** \texttt{table\_linear\_cspline\_1d\_interpolate}
\textbf{Description:} Perform linear interpolation in a linear 1D table.
\textbf{Code lines:} 31
\textbf{Contained by:} module \texttt{tables}

**function:** \texttt{table\_linear\_cspline\_1d\_interpolate\_gradient}
\textbf{Description:} Perform linear interpolation in a linear 1D table.
\textbf{Code lines:} 31
\textbf{Contained by:} module \texttt{tables}

**subroutine:** \texttt{table\_linear\_cspline\_1d\_populate}
\textbf{Description:} Populate a 1-D linear table.
\textbf{Code lines:} 27
\textbf{Contained by:} module \texttt{tables}
\textbf{Modules used:} \texttt{galacticus_error}

**subroutine:** \texttt{table\_linear\_cspline\_1d\_populate\_single}
\textbf{Description:} Populate a single element of a 1-D linear table.
\textbf{Code lines:} 28
**18.1. Program units**

**Contained by:** module *tables*

**Modules used:** galacticus_error

**subroutine:** table_logarithmic_1d_create

*Description:* Create a 1-D logarithmic table.

*Code lines:* 11

**Contained by:** module *tables*

**function:** table_logarithmic_1d_interpolate

*Description:* Perform linear interpolation in a logarithmic 1D table.

*Code lines:* 9

**Contained by:** module *tables*

**function:** table_logarithmic_1d_interpolate_gradient

*Description:* Perform linear interpolation in a logarithmic 1D table.

*Code lines:* 9

**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:* 11

**Contained by:** module *tables*

**function:** table_logarithmic_cspline_1d_interpolate

*Description:* Perform linear interpolation in a logarithmic 1D table.

*Code lines:* 9

**Contained by:** module *tables*

**function:** table_logarithmic_cspline_1d_interpolate_gradient

*Description:* Perform linear interpolation in a logarithmic 1D table.

*Code lines:* 9

**Contained by:** module *tables*

**function:** table_logarithmic_cspline_1d_x

*Description:* Return the $i$th $x$-value for a logarithmic 1D table.

*Code lines:* 8

**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:** `power_spectra.tasks.F90`

**Description:** Contains a module which implements calculations of power spectra and related properties for output.

**Code lines:** 162

**module:** `power_spectrum_tasks`

**Description:** Implements calculations of power spectra and related properties for output.

**Code lines:** 142

**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:** 73

**Contained by:** module `power_spectrum_tasks`

**Modules used:** `cosmological_parameters`, `input_parameters`, `memory_management`, `numerical_constants_math`, `numerical_ranges`, `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`, `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:** 122

**module:** `radiation_cmb`

**Description:** Implements a cosmic microwave background radiation component.

**Code lines:** 102

**Contained by:** file `radiation.cosmic_microwave_background.F90`
18.1. Program units

**Used by:**
- function `radiation_flux`
- subroutine `radiation_set`
- 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:* 22
*Contained by:* module `radiation_cmb`
*Modules used:* `cosmology_functions` `galacticus_nodes` `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:* 182

**module: radiation_intergalactic_background**
*Description:* Implements an intergalactic background (excluding the CMB) radiation component.
*Code lines:* 162
*Contained by:* file `radiation.intergalactic_background.F90`
*Modules used:* `galacticus_nodes`
*Used by:* function `radiation_flux` subroutine `radiation_set` 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:* 52
*Contained by:* module `radiation_intergalactic_background`
*Modules used:* `galacticus_error` `input_parameters` `iso_varying_string` `radiation_igb_file`

**subroutine: radiation_set_intergalactic_background**
*Description:* Property setting routine for the radiation component from file method.
*Code lines:* 17
18. Source Code Documentation

**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:* 202

**module:** `radiation_igb_file`

*Description:* Implements an intergalactic background radiation component read from a file.

*Code lines:* 182

*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:* `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:* 87

*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:* 17

*Contained by:* module `radiation_igb_file`

*Modules used:* `galacticus_nodes memory_management`

**file:** `radiation.null.F90`
18.1. Program units

module: radiation_null
  Code lines: 57
  Contained by: file radiation_null.F90
  Used by: function radiation_flux
  Subroutine radiation_set
  Description: Flux method for the null radiation component.
  Code lines: 10
  Contained by: module radiation_null

subroutine: radiation_flux_null
  Description: Property setting routine for null radiation component.
  Code lines: 9
  Contained by: module radiation_null

subroutine: radiation_set_null
  Description: Temperature method for the null radiation component.
  Code lines: 9
  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

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: 111

module: ram_pressure_stripping_mass_loss_rate_disks_simple

Description: Implements simple mass loss rates from disks due to ram pressure stripping.
Code lines: 91
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
Computes the mass loss rate from disks due to ram pressure stripping assuming a simple model. Specifically, the mass loss rate is

\[ \dot{M}_{\text{gas}} = -\alpha M_{\text{gas}} / \tau_{\text{disk}}, \]  

(18.10)

where

\[ \alpha = F_{\text{ram}} / F_{\text{gravity}}, \]  

(18.11)

\(F_{\text{ram}}\) is the ram pressure force from the hot halo (see §12.17), and

\[ F_{\text{gravity}} = 2\pi G \Sigma_{\text{gas}}(r_{1/2}) \Sigma_{\text{total}}(r_{1/2}) \]  

(18.12)

is the gravitational restoring force in the disk at the half-mass radius, \(r_{1/2}\).
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`

Description: Contains a module which implements calculations of satellite merging times using the Boylan-Kolchin et al. [2008] method.
Code lines: 82

module: `dynamical_friction_boylankolchin2008`
Description: Implements calculations of satellite merging times using the Boylan-Kolchin et al. [2008] method.
Code lines: 62
Used by: subroutine `satellite_merging_timescales_initialize`

function: `satellite_time_until_merging_boylankolchin2008`
Description: Return the timescale for merging satellites using the Boylan-Kolchin et al. [2008] method.
Code lines: 38
Contained by: module `dynamical_friction_boylankolchin2008`
Modules used: `dark_matter_halo_scales`, `dark_matter_profiles`, `dynamical_friction_timescale_utilities`, `galacticus_nodes`, `kepler_orbits`, `satellite_orbits`

subroutine: `satellite_time_until_merging_boylankolchin2008_initialize`
Description: Determine if this method is to be used and set pointer appropriately.
Code lines: 9
Contained by: module `dynamical_friction_boylankolchin2008`
Modules used: `iso_varying_string`

Description: Contains a module which implements calculations of satellite merging times using the Jiang et al. [2008] method.
Code lines: 83

module: `dynamical_friction_jiang2008`
Description: Implements calculations of satellite merging times using the Jiang et al. [2008] method.
Code lines: 63
Used by: subroutine `satellite_merging_timescales_initialize`

function: `satellite_time_until_merging_jiang2008`
Description: Return the timescale for merging satellites using the Jiang et al. [2008] method.
Code lines: 39
Contained by: module `dynamical_friction_jiang2008`
18.1. Program units

Modules used:  
dark_matter_halo_scales  
dark_matter_profiles  
dynamical_friction_timescale_-
utilities  
galacticus_nodes  
satellite_orbits

subroutine: satellite_time_until_merging_jiang2008Initialize

Description:  
Determine if this method is to be used and set pointer appropriately.

Code lines:  
9

Contained by:  
module dynamical_friction_jiang2008

Modules used:
iso_varying_string

file: satellites.merging.dynamical_friction.timescale.Lacey-Cole.F90

Description:  
Contains a module which implements calculations of satellite merging times using the Lacey and Cole [1993] method.

Code lines:
80

module: dynamical_friction_lacey_cole

Description:  

Code lines:
60

Contained by:  
file satellites.merging.dynamical_friction.timescale.Lacey-Cole.F90

Used by:  
subroutine satellite_merging timescales_initialize

function: satellite_time_until_merging_lacey_cole

Description:  
Return the timescale for merging satellites using the Lacey and Cole [1993] method.

Code lines:
36

Contained by:  
module dynamical_friction_lacey_cole

Modules used:
dark_matter_halo_scales  
dynamical_friction_timescale_-
utilities  
galacticus_nodes  
kepler_orbits

subroutine: satellite_time_until_merging_lacey_coleInitialize

Description:  
Determine if this method is to be used and set pointer appropriately.

Code lines:
9

Contained by:  
module dynamical_friction_lacey_cole

Modules used:
iso_varying_string

file: satellites.merging.dynamical_friction.timescale.Lacey-Cole_Tormen.F90

Description:  
Contains a module which 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:
130

module: dynamical_friction_lacey_cole_tormen
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: 108

Contained by: file satellites.merging.dynamical_friction.timescale.Lacey-Cole_Tormen.F90

Modules used: fgs1

Used by: subroutine galacticus_state_retrieve subroutine galacticus_state_snapshot
subroutine galacticus_state_store subroutine satellite_merging_-timescales_initialize

function: satellite_time_until_merging_lacey_cole_tormen
Description: Return the timescale for merging satellites using the Lacey and Cole [1993] method.
Code lines: 36
Contained by: module dynamical_friction_lacey_cole_tormen
Modules used: dark_matter_halo_scales dynamical_friction_timescale_-utilities
galacticus_nodes gaussian_random
kepler_orbits

subroutine: satellite_time_until_merging_lacey_cole_tormen_initialize
Description: Determine if this method is to be used and set pointer appropriately.
Code lines: 9
Contained by: module dynamical_friction_lacey_cole_tormen
Modules used: iso_varying_string

subroutine: satellite_time_until_merging_lacey_cole_tormen_snapshot
Description: Store a snapshot of the random number generator internal state.
Code lines: 7
Contained by: module dynamical_friction_lacey_cole_tormen

subroutine: satellite_time_until_merging_lacey_cole_tormen_state_retrieve
Description: Write the stored snapshot of the random number state to file.
Code lines: 10
Contained by: module dynamical_friction_lacey_cole_tormen
Modules used: pseudo_random

subroutine: satellite_time_until_merging_lacey_cole_tormen_state_store
Description: Write the stored snapshot of the random number state to file.
Code lines: 10
Contained by: module dynamical_friction_lacey_cole_tormen
Modules used: pseudo_random

file: satellites.merging.dynamical_friction.timescale.Wetzel-White.F90
Description: Contains a module which implements calculations of satellite merging times using the Wetzel and White [2010] method.
Code lines: 69

module: dynamical_friction_wetzel_white
Description: Implements calculations of satellite merging times using the Wetzel and White [2010] method.
Code lines: 47
Contained by: file satellites.merging.dynamical_friction.timescale.Wetzel-White.F90
Used by: subroutine satellite_merging_-timescales_initialize

function: satellite_time_until_merging_wetzel_white
18. Source Code Documentation

Description: Return the timescale for merging satellites using the Wetzel and White [2010] method.
Code lines: 23
Contained by: module dynamical_friction_wetzel_white
Modules used: cosmology_functions dynamical_friction_timescale_utilities galacticus_nodes kepler_orbits

subroutine: satellite_time_until_merging_wetzel_white_initialize
Description: Determine if this method is to be used and set pointer appropriately.
Code lines: 9
Contained by: module dynamical_friction_wetzel_white
Modules used: iso_varying_string

file: satellites.merging.dynamical_friction.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.dynamical_friction.timescale.utilities.F90
Used by: function satellite_time_until_merging_boylankolchin2008 function satellite_time_until_merging_jiang2008
function satellite_time_until_merging_lacey_cole function satellite_time_until_merging_lacey_cole_tormen
function satellite_time_until_merging_wetzel_white

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 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.
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_disk_exponential_satellite_merging
subroutine node_component_inter_output_standard_satellite_merger
subroutine node_component_spheroid_standard_satellite_merging
subroutine satellite_merging_mass_movement_store
module satellite_merging_mass_movements_baugh2005
module satellite_merging_mass_movements_very_simple
subroutine satellite_merging_mass_movement_store
module satellite_merging_mass_movements_simple
function half_mass_radius_root_cole2000
subroutine satellite_merging_remnant_progenitor_properties_cole2000

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_simple
subroutine satellite_merging_mass_movements_simple_initialize
subroutine satellite_merging_mass_movements_simple

subroutine: satellite_merging_mass_movement_simple
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_movements_simple_initialize
Description: Test if this method is to be used and set procedure pointer appropriately.
Code lines: 44
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_very_simple.descriptors`

*Used by:* subroutine `satellite_merging_mass_movement.very_simple`

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.preset.F90**

*Description:* Contains a module which implements calculations of satellite merging times using preset values.

*Code lines:* 57

module: **satellite_merging_times_preset**

*Description:* Implements calculations of satellite merging times using preset values.

*Code lines:* 37

*Contained by:* file `satellites.merging.preset.F90`

*Used by:* subroutine `satellite_merging_timescales.initialize`

function: **satellite_time_until_merging.preset**

*Description:* Return the timescale for merging satellites using the preset value.

*Code lines:* 13

*Contained by:* module `satellite_merging_times_preset`

*Modules used:* `galacticus_nodes` `kepler_orbits`

subroutine: **satellite_time_until_merging.preset.initialize**

*Description:* Determine if this method is to be used and set pointer appropriately.

*Code lines:* 9

*Contained by:* module `satellite_merging_times_preset`

*Modules used:* `iso_varying_string`


*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
Used by: subroutine satellite_merging_remnant_size_cole2000

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
galacticus_display
galacticus_nodes
numerical_comparison
satellite_merging_remnant_sizes_progenitors
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: 217

module: satellite_merging_remnant_sizes_covington2008
Description: Implements the Covington et al. [2008] algorithm for merger remnant sizes.
Code lines: 197
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: 139
Contained by: module satellite_merging_remnant_sizes_covington2008
Modules used:
  dark_matter_halo_scales
galacticus_error
galacticus_display
iso_varying_string
numerical_comparison
numerical_constants_physical
satellite_merging_remnant_sizes_progenitors
18.1. Program units

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
Contained by: file satellites.merging.remnant_sizes.F90
Modules used: iso_varying_string
Used 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_- satellite_merging_remnant_sizes_null
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.
18. Source Code Documentation

**Code lines:** 9
**Contained by:** module satellite_merging_remnant_sizes_null
**Modules used:** iso_varying_string

**file:** satellites.merging.remnant_sizes.progenitor_properties.Cole2000.F90
**Description:** Contains a module which implements calculations of progenitor properties for merger remnant calculations using the algorithm of Cole et al. [2000].
**Code lines:** 258

**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:** 237
**Contained by:** file satellites.merging.remnant_sizes.progenitor_properties.Cole2000.F90
**Modules used:** galacticus_nodes root_finder
**Used by:** subroutine satellite_merging_remnant_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:** 14
**Contained by:** module satellite_merging_remnant_progenitors_properties_cole2000
**Modules used:** 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
18.1. Program units

**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`  
- `satellite_merging_mass_movements_descriptors`

**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*: 211  
*Contained by*: module `satellite_merging_remnant_progenitors_properties_standard`  
*Modules used*:  
- `galacticus_error`  
- `galacticus_nodes`  
- `iso_varying_string`

**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*: 15  
*Contained by*: module `satellite_merging_remnant_progenitors_properties_standard`  
*Modules used*:  
- `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

file: satellites.merging.timescale.F90

Description: Contains a module that implements calculations of merging timescales for satellites.

Code lines: 120

module: satellite_merging_timescales

Description: Implements calculations of merging timescales for satellites.

Code lines: 100

Contained by: file satellites.merging.timescale.F90

Modules used: iso_varying_string

Used by: subroutine node_component_satellite_standard_create
        subroutine node_component_satellite_standard_virial_orbit_set
        subroutine node_component_satellite_very_simple_create

subroutine: satellite_merging_timescales_initialize

Description: Initialize the satellite merging timescale module.

Code lines: 64

Contained by: module satellite_merging_timescales

Modules used: dynamical_friction_boylankolchin2008 dynamical_friction_jiang2008
dynamical_friction_lacey_cole dynamical_friction_lacey_cole_tormen
dynamical_friction_wetzel_white galacticus_error
input_parameters satellite_merging_times_preset

function: satellite_time_until_merging

Description: Return the satellite merging timescale for thisNode (in units of Gyr).

Code lines: 14

Contained by: module satellite_merging_timescales

Modules used: galacticus_nodes kepler_orbits


Description: Contains a module which implements the Benson [2005] orbital parameter distribution for merging subhalos.

Code lines: 142

module: virial_orbits_benson2005

Description: Implements the Benson [2005] orbital parameter distribution for merging subhalos.

Code lines: 122

18.1. Program units

**Modules used:** \texttt{fgsl}

**Used by:**
- subroutine \texttt{galacticus_state_retrieve}
- subroutine \texttt{galacticus_state_snapshot}
- subroutine \texttt{galacticus_state_store}
- function \texttt{virial_orbital_parameters}

**function:** \texttt{virial\_orbital\_parameters\_benson2005}

**Description:**
Return orbital parameters of a satellite selected at random from the fitting function found by Benson [2005].

**Code lines:** 52

**Contained by:** module \texttt{virial\_orbits\_benson2005}

**Modules used:**
- \texttt{dark\_matter\_halo\_scales}
- \texttt{galacticus\_error}
- \texttt{galacticus\_nodes}
- \texttt{kepler\_orbits}
- \texttt{pseudo\_random}

**subroutine:** \texttt{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 \texttt{virial\_orbits\_benson2005}

**Modules used:** \texttt{iso\_varying\_string}

**subroutine:** \texttt{virial\_orbital\_parameters\_benson2005\_snapshot}

**Description:**
Store a snapshot of the random number generator internal state.

**Code lines:** 7

**Contained by:** module \texttt{virial\_orbits\_benson2005}

**subroutine:** \texttt{virial\_orbital\_parameters\_benson2005\_state\_retrieve}

**Description:**
Write the stored snapshot of the random number state to file.

**Code lines:** 10

**Contained by:** module \texttt{virial\_orbits\_benson2005}

**Modules used:** \texttt{pseudo\_random}

**subroutine:** \texttt{virial\_orbital\_parameters\_benson2005\_state\_store}

**Description:**
Write the stored snapshot of the random number state to file.

**Code lines:** 10

**Contained by:** module \texttt{virial\_orbits\_benson2005}

**Modules used:** \texttt{pseudo\_random}

**file:** \texttt{satellites.merging.virial\_orbits.F90}

**Description:**
Contains a module which implements satellite orbital parameters at virial radius crossing.

**Code lines:** 105

**module:** \texttt{virial\_orbits}

**Description:**
Implements satellite orbital parameters at virial radius crossing.

**Code lines:** 85

**Contained by:** file \texttt{satellites.merging.virial\_orbits.F90}

**Modules used:**
- \texttt{galacticus\_nodes}
- \texttt{iso\_varying\_string}
- \texttt{scan\_for\_mergers}
- \texttt{node\_component\_satellite\_standard\_create}
- \texttt{node\_component\_satellite\_standard\_virial\_orbit}

**Used by:**
- subroutine \texttt{scan\_for\_mergers}
- subroutine \texttt{node\_component\_satellite\_standard\_create}
- function \texttt{node\_component\_satellite\_standard\_virial\_orbit}
**function:** virial_orbital_parameters  
*Description:* Returns virial orbital parameters.  
*Code lines:* 64  
*Contained by:* module virial_orbits  
*Modules used:* galacticus_error  
kepler_orbits  
virial_orbits_fixed  
input_parameters  
virial_orbits_benson2005  
virial_orbits_wetzel2010

**file:** satellites.merging.virial_orbits.Wetzel2010.F90  
*Description:* Contains a module which implements the Wetzel [2010] orbital parameter distribution for merging subhalos.  
*Code lines:* 236

**module:** virial_orbits_wetzel2010  
*Description:* Implements the Wetzel [2010] orbital parameter distribution for merging subhalos.  
*Code lines:* 216  
*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  
*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:* 87  
*Contained by:* module virial_orbits_wetzel2010  
*Modules used:* cosmology_functions  
dark_matter_halo_scales  
kepler_orbits  
root_finder  
critical_overdensity  
galacticus_nodes  
pseudo_random

**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
18.1. Program units

**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:* 97

**module: virial_orbits_fixed**
*Description:* Implements a fixed orbital parameter distribution for merging subhalos.
*Code lines:* 77
*Contained by:* file satellites.merging.virial_orbits.fixed.F90
*Used by:* function virial_orbital_parameters

**function: virial_orbital_parameters_fixed**
*Description:* Return fixed orbital parameters for a satellite.
*Code lines:* 23
*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:* 155

**module: satellite_orbits**
*Description:* Implements calculations related to satellite orbits.
*Code lines:* 135
18. Source Code Documentation

**function: equivalent_circular_orbit_solver**
*Description:* Root function used in finding equivalent circular orbits.
*Code lines:* 8
*Contained by:* module satellite_orbits
*Modules used:* dark_matter_profiles

**function: pericenter_solver**
*Description:* Root function used in finding orbital pericentric radius.
*Code lines:* 8
*Contained by:* module satellite_orbits
*Modules used:* dark_matter_profiles

**function: satellite_orbit_convert_to_current_potential**
*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:

$$ e \rightarrow e + V_{\text{vir},0}^2 + \Phi(r_{\text{vir},0}), $$

where subscript 0 refers to the original halo in which the orbit was defined and $\Phi(r)$ is the potential of the current halo.
*Code lines:* 26
*Contained by:* module satellite_orbits
*Modules used:* galactic_structure_potentials kepler_orbits numerical_constants_physical

**function: satellite_orbit_equivalent_circular_orbit_radius**
*Description:* Solves for the equivalent circular orbit radius for thisOrbit in hostNode.
*Code lines:* 30
*Contained by:* module satellite_orbits
*Modules used:* dark_matter_halo_scales kepler_orbits root_finder

**subroutine: satellite_orbit_pericenter_phase_space_coordinates**
*Description:* Solves for the pericentric radius and velocity of thisOrbit in hostNode.
*Code lines:* 38
*Contained by:* module satellite_orbits
*Modules used:* kepler_orbits root_finder

**file: satellites.promotion.F90**
*Description:* Contains a module which handles events where a satellite is moved to a new host halo.
18.1. Program units

**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

**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**: star_formation.IMF.Baugh2005TopHeavy.F90
- **Description**: Contains a module which implements the top-heavy stellar initial mass function from Baugh et al. [2005].
- **Code lines**: 230

**module**: star_formation_imf_baugh2005topheavy
- **Description**: Implements the top-heavy stellar initial mass function from Baugh et al. [2005].
- **Code lines**: 210
- **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
- **Modules used**: input_parameters, star_formation_imf_ppl

**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
  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: 239

module: star_formation_imf_chabrier
  Description: Implements the Chabrier stellar initial mass function.
  Code lines: 219
  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
18.1. Program units

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: 1684

module: star_formation_imf
Description: Implements functionality related to the stellar initial mass function.
18. Source Code Documentation

**Code lines:** 1664  
**Contained by:** file `star_formation.IMF.F90`  
**Modules used:**
- `abundances_structure`
- `iso_varying_string`
- `star_formation_imf_baugh2005topheavy`
- `star_formation_imf_chabrier`
- `star_formation_imf_kennicutt`
- `star_formation_imf_piecewisepowerlaw`
- `star_formation_imf_salpeter`
- `star_formation_imf_scalo`
- `star_formation_imf_select_disk_spheroid`
- `star_formation_imf_select_fixed`

**Used by:**
- function `stellar_population_luminosity`
- subroutine `stellar_population_properties_rates_instantaneous`
- subroutine `stellar_population_properties_rates_noninstantaneous`
- 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 `starFormationRate` and `fuelAbundances`. The energy input rate is computed for the given `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:** 303  
**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: 365
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_minimum_mass

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_fraction_instantaneous

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: 311
Contained by: module star_formation_imf
Modules used: dates_and_times file_utilities
               fox_dom fox_wxml
               galacticus_display galacticus_error
18.1. Program units

```plaintext
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 given 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
              supernovae_type_ia

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: 146
Contained by: module star_formation_imf
Modules used: galacticus_error    input_parameters
              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].
```
module: star_formation_imf_kennicutt
Description: Implements the Kennicutt stellar initial mass function.
Code lines: 230
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
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_ppl

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
18.1. Program units

**Contained by:** module `star_formation_imf_kennicutt`

**Modules used:** `star_formation_imf_ppl`  `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:** 230

**module:** `star_formation_imf_kroupa`

**Description:** Implements the Kroupa stellar initial mass function.

**Code lines:** 210

**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
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

subroutine star_formation_imf_yield_instantaneous_kroupa
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation_imf_kroupa

file: 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: 230

module star_formation_imf_millerscalo
Description: Implements the MillerScalo stellar initial mass function.
Code lines: 210
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
18.1. Program units

**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:** 230

**module:** `star_formation_imf_salpeter`

**Description:** Implements the Salpeter stellar initial mass function.

**Code lines:** 210

**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.
18. Source Code Documentation

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
  *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 tables

**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:* 230

**module:** star_formation_imf_scalo
  *Description:* Implements the Scalo stellar initial mass function.
  *Code lines:* 210
  *Contained by:* file star_formation.IMF.Scalo.F90
18.1. Program units

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 star_formation_imf_ppl

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: 299

module: star_formation.imf.piecewise_power_law
Description: Implements an arbitrary piecewise power-law stellar initial mass function.
Code lines: 279
Contained by: file star_formation.IMF.piecewise_power_law.F90
Used by: module star_formation.imf

subroutine: star_formation.imf.initialize.piecewise_power_law
Description: Initialize the PiecewisePowerLaw IMF module.
Code lines: 98
Contained by: module star_formation.imf.piecewise_power_law
Modules used: galacticus_error input_parameters
memory_management star_formation.imf.ppl

subroutine: star_formation.imf.maximum_mass.piecewise_power_law
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation.imf.piecewise_power_law

subroutine: star_formation.imf.minimum_mass.piecewise_power_law
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation.imf.piecewise_power_law

subroutine: star_formation.imf.phi.piecewise_power_law
Description: Register the name of this IMF.
Code lines: 15
Contained by: module star_formation.imf.piecewise_power_law
Modules used: star_formation.imf.ppl

subroutine: star_formation.imf.recycled.instantaneous.piecewise_power_law
Description: Register the name of this IMF.
Code lines: 13
Contained by: module star_formation.imf.piecewise_power_law

subroutine: star_formation.imf.register_name.piecewise_power_law
Description: Register the name of this IMF.
Code lines: 22
Contained by: module star_formation.imf.piecewise_power_law
Modules used: iso_varying_string

subroutine: star_formation.imf.register.piecewise_power_law
Description: Register this IMF by incrementing the count and keeping a record of the assigned index.
18.1. Program units

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
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

file: star_formation.IMF.utilities.F90
  Description: Contains a module of useful utilities required by the IMF subsystem.
  Code lines: 44

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_-
           subroutine imf_select_fixed_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 func-
                tions.
  Code lines: 102

module: star_formation_imf_ppl
  Description: Implement 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_-
            subroutine star_formation_imf_phi_-
            initialize_kroupa
            baugh2005topheavy
            tabulate_baugh2005topheavy
            kroupa
            baugh2005topheavy
            initialize_kennicutt
            tabulate_kennicutt
            kroupa
18.1. Program units

subroutine star_formation_imf_-_tabulate_kroupa
subroutine star_formation_imf_phi_-_millerscalo
subroutine star_formation_imf_-_initialize_salpeter
subroutine star_formation_imf_-_initialize_piecewisepowerlaw

subroutine star_formation_imf_-_initialize_millerscalo
subroutine star_formation_imf_phi_-_millerscalo
subroutine star_formation_imf_-_tabulate_millerscalo
subroutine star_formation_imf_-_initialize_salpeter
subroutine star_formation_imf_-_tabulate_salpeter
subroutine star_formation_imf_phi_scalo
subroutine star_formation_imf_-_initialize_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_disk_outflow_rate_creasey2012

function: star_formation_feedback_disk_outflow_rate_creasey2012
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

$$\dot{M}_{\text{outflow}} = \beta_0 \Sigma_{g,1}(r) f_\text{g}(r) \bar{\Sigma}_\star(r) 2\pi r dr,$$

where $\Sigma_{g,1}(r)$ is the surface density of gas in units of $M_\odot \, \text{pc}^{-2}$, $f_\text{g}(r)$ is the gas fraction, $\bar{\Sigma}_\star(r)$ is the surface density of star formation rate, $\beta_0 = \text{[starFormationFeedbackDisksCreasy2012Beta0]}$, $\mu = \text{[starFormationFeedbackDisksCreasy2012Mu]}$, and $\nu = \text{[starFormationFeedbackDisksCreasy2012Nu]}$. 

**function: star_formation_feedback_disk_outflow_rate_creasey2012_integrand**

**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:**

- iso_c_binding
- numerical_constants_math
- numerical_integration
- stellar_feedback

**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**
18.1. Program units

**function:** starformation feedback disk outflow rate template

- Code lines: 6
- Contained by: module starformation feedback disks

**subroutine:** starformation feedback disks initialize

- Description: Initialize the disk star formation feedback module.
- Code lines: 59
- Contained by: module starformation feedback disks
- Modules used: galacticus error input parameters
  starformation feedback disks creasey2012
  starformation feedback disks fixed
  starformation feedback disks_fixed
  starformation feedback disks fixed scaling
  starformation feedback disks fixed law

**file:** starformation 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
- Contained by: module starformation feedback disks
- Modules used: galacticus nodes

**module:** starformation feedback disks fixed

- Description: Implements a fixed outflow rate due to star formation feedback in galactic disks.
- Code lines: 54
- Contained by: file starformation feedback disks fixed F90
- Modules used: galacticus nodes

**function:** starformation feedback disk outflow rate fixed

- Description: Returns the outflow rate (in \(M_\odot\) 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 starformation feedback disks fixed
- Modules used: stellar feedback

**subroutine:** starformation feedback disks fixed initialize

- Description: Initializes the “fixed” disk star formation feedback module.
- Code lines: 25
- Contained by: module starformation feedback disks fixed
- Modules used: input parameters iso varying string

**file:** starformation 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: 112

**module:** starformation 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.
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.

subroutine: star_formation_feedback_disks_halo_scaling_initialize
Description: Initializes the “halo scaling” disk star formation feedback module.

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.

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

$$M_{\text{outflow}} = \left( \frac{V_{\text{disk, outflow}}}{V_{\text{disk}}} \right)^{\alpha_{\text{disk, outflow}}}, \quad (18.15)$$

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.
18.1. Program units

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 $M_\odot \text{Gyr}^{-1}$) for star formation in the galactic spheroid of thisNode. The outflow rate is given by

$$\dot{M}_{\text{outflow}} = \left( \frac{V_{\text{spheroid, outflow}}}{V_{\text{spheroid}}} \right)^{\alpha_{\text{spheroid, outflow}}},$$

(18.16)

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

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_expulsive_feedback_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_expulsive_feedback_disks_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_expulsive_feedback_disks_null
Modules used: galacticus_nodes

subroutine: star_formation_expulsive_feedback_disks_null_initialize

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 \( M_\odot \text{ Gyr}^{-1} \)) for star formation in the galactic disk of thisNode. The outflow rate is given by

\[
\dot{M}_{\text{outflow}} = f_{SW,0} \begin{cases} 
1 & \text{if } V_{\text{disk}} < V_{\text{disk,SW}}, \\
(V_{\text{disk,SW}}/V_{\text{disk}})^2 & \text{if } V_{\text{disk}} \geq V_{\text{disk,SW}},
\end{cases}
\] (18.17)

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
18.1. Program units

**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`

**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 this Node.

**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_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_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
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

$$M_{\text{outflow}} = \begin{cases} 
1 & \text{if } V_{\text{spheroid}} < V_{\text{spheroid,SW}}, \\
(V_{\text{spheroid,SW}}/V_{\text{spheroid}})^2 & \text{if } V_{\text{spheroid}} \geq V_{\text{spheroid,SW}} 
\end{cases},$$

(18.18)

where $V_{\text{spheroid,SW}} = [\text{spheroidSuperwindVelocity}]$ and $f_{\text{SW,0}} = [\text{spheroidSuperwindMassLoading}]$. 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.
18.1. Program units

**Code lines:** 200  
**Contained by:** file `star_formation.rate_surface_density.disks.Blitz-Rosolowsky.F90`  
**Modules used:** `galacticus_nodes`  
**Used by:** subroutine `galacticus_calculations_reset`  
**Description:** Returns the star formation rate surface density (in $\text{M}_\odot \text{Gyr}^{-1} \text{Mpc}^{-2}$) 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`  
**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`  
**Description:** Reset the extended Schmidt relation calculation.

**Code lines:** 114  
**Contained by:** module `star_formation_rate_surface_density_disks`  
**Description:** Implements calculations of star formation rate surface densities for galactic disks.

**Code lines:** 12  
**Contained by:** module `star_formation_rate_surface_density_disks`  
**Description:** Returns the star formation rate surface density (in $\text{M}_\odot \text{Gyr}^{-1} \text{Mpc}^{-2}$) in the disk component of `thisNode` at the given radius.

**Code lines:** 1207
18. Source Code Documentation

**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

**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 \( M_\odot \text{ Gyr}^{-1} \text{ Mpc}^{-2} \)) 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:

$$\Sigma_* = A \left( x_H \frac{\Sigma_{\text{gas}}}{M_{\odot} \text{pc}^{-2}} \right)^N,$$

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
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

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]:

$$\dot{\Sigma}_\star = A \left( \frac{x_H \Sigma_{\text{gas}}}{M_\odot \text{pc}^{-2}} \right)^{N_1} \left( \frac{\Sigma_\star}{M_\odot \text{pc}^{-2}} \right)^{N_2} ,$$  \hspace{1cm} (18.21)

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

18.1. Program units

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
Used by: function node_component_disk_exponential_star_formation_rate

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 timecale 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
dynamical_time 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 a dynamical time-based star formation timescale for galactic disks.
Code lines: 131

module: star_formation_timescale_disks_dynamical_time

Description: Implements a a dynamical time-based star formation timescale for galactic disks.
Code lines: 111
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
18. Source Code Documentation

**Description:** Returns the timescale (in Gyr) for star formation in the galactic disk of `thisNode`. The timescale is given by

\[
\tau_* = \epsilon_*^{-1} \tau_{\text{dynamical, disk}} \left( \frac{V_{\text{disk}}}{200 \text{km/s}} \right)^{\alpha_*},
\]

where \( \epsilon_* (= \text{starFormationDiskEfficiency}) \) is a star formation efficiency and \( \alpha_* (= \text{starFormationDiskVelocityExponent}) \) controls the scaling with velocity. Note that \( \tau_{\text{dynamical, disk}} = R_{\text{disk}} / V_{\text{disk}} \) 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`
18.1. Program units

**Description:** Initializes the “dynamical time” disk star formation timescale module.

**Code lines:** 55

**Contained by:** module `star_formation_timescale_disks_dynamical_time`

**Modules used:** `galacticus_error`  `galacticus_nodes`  `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.

**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_fixed_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 timescale.

**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:** 146

**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:** 125

**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:** 33

**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: 49
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 a 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

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.
18.1. Program units

module: star_formation_timescales_spheroids

Description: Implements calculations of star formation timescales for galactic spheroids.

Code lines: 114

Contained by: file star_formation.timescales.spheroids.F90

Modules used: galacticus_nodes iso_varying_string

Used by: function node_component_spheroid_standard_star_formation_rate

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

Modules used: galacticus_error input_parameters

star_formation_timescale_spheroids_dynamical_time

dynamical_time

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_spheroid_dynamical_time
18. Source Code Documentation

**Description:** Returns the timescale (in Gyr) for star formation in the galactic spheroid of `thisNode`. The timescale is given by

\[
\tau_\star = \epsilon^{-1}_\star \tau_{\text{dynamical,spheroid}} \left( \frac{V_{\text{spheroid}}}{200 \text{ km/s}} \right)^{\alpha_\star},
\]

(18.23)

where \( \epsilon_\star = \text{starFormationSpheroidEfficiency} \) is a star formation efficiency and \( \alpha_\star = \text{starFormationSpheroidVelocityExponent} \) controls the scaling with velocity. Note that \( \tau_{\text{dynamical,spheroid}} = \frac{R_{\text{spheroid}}}{V_{\text{spheroid}}} \) 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`

**Modules used:** `galacticus_nodes` `numerical_constants_astronomical`

**subroutine:** `star_formation_timescale_spheroids_dynamical_time_initialize`
18.1. Program units

**Description:** Initializes the “dynamical time” spheroid star formation timescale module.

**Code lines:** 49

**Contained by:** module `star_formation_timescale_spheroids_dynamical_time`

**Modules used:** `input_parameters` `iso_varying_string`

**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:** `metal_yield_integrand` `recycled_fraction_integrand`

`star_is_evolved` `stellar_feedback_cumulative_energy_input_standard`

`snepopii_cumulative_energy_hegerwoosley` `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` `input_parameters`

`stellar_astrophysics_file`

**file:** `stellar_astrophysics.feedback.F90`

**Description:** Contains a module which provides calculations of stellar feedback.
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 star_formation_feedback_disk_outflow_rate_creasey2012
function star_formation_feedback_disk_outflow_rate_halo_scaling
function star_formation_feedback_disk_outflow_rate_power_law
function star_formation_feedback_spheroid_outflow_rate_power_law
function star_formation_expulsive_feedback_spheroid_outflow_rate_sw
subroutine stellar_population_properties_rates_instantaneous

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
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
18.1. Program units

**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_-

  *initalize*

**function: star_ejected_mass_file**

*Description:* Return the mass ejected during the lifetime of a star of given \texttt{initialMass} and \texttt{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 \texttt{lifetime} and \texttt{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 \texttt{initialMass} and \texttt{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 \texttt{initialMass} and \texttt{metallicity}.

*Code lines:* 21

*Contained by:* module stellar_astrophysics_file

**function: stellar_astrophysics_file_format_version**
18. Source Code Documentation

**subroutine: stellar_astrophyics_file_initialize**

*Description:* Initialize the stellar astrophysics module.

*Code lines:* 196

*Contained by:* module stellar_astrophyics_file

*Modules used:*
- atomic_data
- fox_dom
- galacticus_error
- galacticus_input_paths
- input_parameters
- io_xml
- iso_varying_string
- memory_management

**file: stellar_astrophyics.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_astrophyics.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.

*Code lines:* 11

*Contained by:* module supernovae_population_iii

**subroutine: supernovae_population_iii_initialize**

*Description:* Initialize the Population III supernovae module.

*Code lines:* 52

*Contained by:* module supernovae_population_iii

*Modules used:* galacticus_error

**file: stellar_astrophyics.supernovae_PopulationIII.HegerWoosley.F90**

*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

*Contained by:* file stellar_astrophyics.supernovae_PopulationIII.HegerWoosley.F90

*Used by:* subroutine supernovae_population_iii_initialize
function: snepopiii_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
Contained by: module supernovae_population_iii_hegerwoosley
Modules used: fgs1 numerical_interpolation stellar_astrophysics

subroutine: supernovae_population_iii_hegerwoosley_initialize
Description: Initialize the “Heger-Woosley2002” Population III supernovae module.
Code lines: 41
Contained by: module supernovae_population_iii_hegerwoosley
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
Modules used: iso_varying_string
Used by: function metal_yield_integrand function stellar_feedback_cumulative_energy_input_standard

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
Modules used: galacticus_error input_parameters supernovae_type_ia_nagashima

18.1. Program units
### 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

**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
18.1. Program units

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: 331

module: stellar_astrophysics_tracks_file
Description: Implements stellar tracks.
Code lines: 311
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: 19
Contained by: module stellar_astrophysics_tracks_file

function: stellar_luminosity_file
Description: Return the bolometric luminosity (in $L_\odot$) for a star of given initialMass, metallicity and age.
Code lines: 19
Contained by: module stellar_astrophysics_tracks_file

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
input_parameters
io_hdf5

1223
function: stellar_tracks_interpolation_do
Description: Using precomputed factors, interpolate in metallicity, mass and age in the given stellarTracks.
Code lines: 20
Contained by: module stellarAstrophysics_tracks_file

subroutine: stellar_tracks_interpolation_get
Description: Get interpolating factors for stellar tracks.
Code lines: 86
Contained by: module stellarAstrophysics_tracks_file
Modules used: numerical_interpolation

file: stellarAstrophysics.winds.F90
Description: Contains a module which provides calculations of stellar winds.
Code lines: 124

module: stellarAstrophysics_winds
Description: Provides calculations of stellar winds.
Code lines: 104
Contained by: file stellarAstrophysics.winds.F90
Modules used: iso_varying_string
Used by: function wind_energy_integrand

subroutine: stellar_winds_initialize
Description: Initialize the stellar winds module.
Code lines: 52
Contained by: module stellarAstrophysics_winds
Modules used: galacticus_error input_parameters

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 stellarAstrophysics_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 stellarAstrophysics_winds

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
18.1. Program units

**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:* 286

**module: stellar_population_luminosities**
*Description:* Implements calculations of stellar population luminosities in the AB magnitude system.
*Code lines:* 266
*Contained by:* file *stellar_populations.luminosities.F90*
*Modules used:* abundances_structure fgsl iso_c_binding
*Used by:* function *stellar_population_luminosities_get*

**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 stellar_population_spectra_postprocess
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: 189
Contained by: module stellar_population_luminosities
Modules used: galacticus_display galacticus_error
input_parameters instruments_filters
iso_varying_string memory_management
numerical_constants_astronomical numerical_integration
numerical_interpolation star_formation_imf
stellar_population_spectra string_handling

file: stellar_populations.properties.F90
Description: Contains a module which provides support for stellar population properties.
Code lines: 200

module: stellar_population_properties
Description: Provides support for stellar population properties.
Code lines: 180
Contained by: file stellar_populations.properties.F90
Modules used: abundances_structure galacticus_nodes
histories iso_varying_string
Used by: subroutine node_component_disk_- exponential_create subroutine node_component_disk_- exponential_scale_set
subroutine node_component_disk_- exponential_rate_compute
module node_component_spheroid_- standard

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: stellar_population_properties_history_create
Description: Create any history required for storing stellar population properties.
Code lines: 13
18.1. Program units

**Contained by:** module `stellar_population_properties`

**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:* 18

*Contained by:* module `stellar_population_properties`

**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:** `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.

*Code lines:* 148

**module:** `stellar_population_properties_instantaneous`

*Description:* Implements stellar population properties in the instantaneous recycling approximation.

*Code lines:* 128

*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`

**subroutine:** `stellar_population_properties_instantaneous_initialize`

*Description:* Initializes the instantaneous recycling approximation stellar population properties module.

*Code lines:* 21
18. Source Code Documentation

**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:** 54

**Contained by:** module `stellar_population_properties_instantaneous`

**Modules used:** 
- `abundances_structure`
- `galacticus_nodes`
- `histories`
- `star_formation_imf`
- `stellar_feedback`
- `stellar_population_properties_luminosities`

**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.luminosities.F90`

**Description:** Contains a module which provides properties related to luminosities of stellar populations that are being tracked.

**Code lines:** 314

**module:** `stellar_population_properties_luminosities`

**Description:** Provides properties related to luminosities of stellar populations that are being tracked.

**Code lines:** 294

**Contained by:** file `stellar_populations.properties.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_output_tree_half_light`
- subroutine `galacticus_output_tree_half_light_initialize`
- subroutine `galacticus_output_tree_half_light_names`
- subroutine `galacticus_output_tree_rotation_curve_initialize`
- function `diskluminositiesstellar`
- subroutine `node_component_disk_output`
- subroutine `node_component_disk_exponentialInitializer`
- subroutine `node_component_disk_exponential_output_count`
- subroutine `node_component_spheroid_standardInitializer`
- subroutine `node_component_spheroid_standard_output_count`
- function `spheroidluminositiesstellar`
18.1. Program units

```plaintext
subroutine node_component_disk_-exponential_initialize
subroutine node_component_spheroid_-standard_initialize
subroutine stellar_population_-properties_rates_instantaneous
subroutine stellar_population_-properties_rates_noninstantaneous
```

**function: stellar_population_luminosities_count**

*Description:* Return the number of stellar luminosities being tracked.
*Code lines:* 10
*Contained by:* module `stellar_population_properties_luminosities`

**function: stellar_population_luminosities_get**

*Description:* Return the luminosity in each requested band for a stellar population of $1M_\odot$ with the specified `abundancesStellar` and which formed at cosmological `time` with IMF specified by `imfSelected`.
*Code lines:* 21
*Contained by:* module `stellar_population_properties_luminosities`
*Modules used:* `abundances_structure` `stellar_population_luminosities`

**function: stellar_population_luminosities_index**

*Description:* Return the index of an specified entry in the luminosity list given its name.
*Code lines:* 15
*Contained by:* module `stellar_population_properties_luminosities`
*Modules used:* `galacticus_error`

**function: stellar_population_luminosities_name**

*Description:* Return a name for the specified entry in the luminosity list.
*Code lines:* 8
*Contained by:* module `stellar_population_properties_luminosities`

**function: stellar_population_luminosities_output**

*Description:* Return true or false depending on whether `luminosityIndex` should be output at `time`.
*Code lines:* 19
*Contained by:* module `stellar_population_properties_luminosities`
*Modules used:* `galacticus_error`

**function: stellar_population_luminosities_output_count**

*Description:* Return a count of the number of luminosities to be output at `time`.
*Code lines:* 11
*Contained by:* module `stellar_population_properties_luminosities`

**subroutine: stellar_population_properties_luminosities_initialize**

*Code lines:* 174
*Contained by:* module `stellar_population_properties_luminosities`
*Modules used:* `cosmology_functions` `galacticus_error`
`input_parameters` `instruments_filters`
`memory_management` `stellar_population_spectra_-postprocess`
file: stellar_populations.properties.noninstantaneous.F90
Description: Contains a module which implements stellar population properties with noninstantaneous recycling.
Code lines: 248

module: stellar_population_properties_noninstantaneous
Description: Implements stellar population properties with noninstantaneous recycling.
Code lines: 228
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: 83
Contained by: module stellar_population_properties_noninstantaneous
Modules used: abundances_structure
fgsl
galacticus_nodes
histories
numerical_interpolation
star_formation_imf
stellar_population_properties_luminosities

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
18.1. Program units

**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*: 163

**module**: stellar_population_spectra_conroy  
*Description*: Handles stellar spectra using the Conroy et al. [2009] package.  
*Code lines*: 143  
*Contained by*: file stellar_populations.spectra.Conroy_et_al.F90  
*Modules used*: iso_varying_string

**function**: stellar_population_spectra_conroy_get  
*Description*: Return the luminosity (in units of $L_{\odot} \cdot Hz^{-1}$) 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*: 18  
*Contained by*: module stellar_population_spectra_conroy

**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

**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

**file**: stellar_populations.spectra.F90  
*Description*: Contains a module that implements calculations of stellar population spectra.  
*Code lines*: 143

**module**: stellar_population_spectra  
*Description*: Implements calculations of stellar population spectra.
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: 15
  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

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: 310

module: stellar_population_spectra_file
  Description: Reads and interpolates a file of stellar population spectra.
  Code lines: 290
  Contained by: file stellar_populations.spectra.file.F90
  Modules used: fgsl
  Used by: function stellar_population_spectra_file_format_current
  subroutine stellar_population_spectra_file_get
  subroutine stellar_population_spectra_tabulation_conroy
  module stellar_population_spectra

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: 17
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: 40
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: 66
Contained by: module stellar_population_spectra_file
Modules used: abundances_structure galacticus_error
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 that implements postprocessing of stellar population spectra.
Code lines: 190

module: stellar_population_spectra_postprocess
Description: Implements postprocessing of stellar population spectra.
Code lines: 170
Contained by: file stellar_populations.spectra.postprocess.F90
Modules used: iso_varying_string
18. Source Code Documentation

Used by: function filter_luminosity_integrand subroutine stellar_population_properties_luminosities_initialize

**type:** postprocessor

| Code lines: | 2 |
| Contained by: | module stellar_population_spectra_postprocess |

**type:** postprocessors

| Code lines: | 2 |
| 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_index

| Description: | Return the index to the specified postprocessing chain. |
| Code lines: | 105 |
| Contained by: | module stellar_population_spectra_postprocess |

| Modules used: | galacticus_error input_parameters |
| | stellar_population_spectra-postprocessing_identity stellar_population_spectra-postprocessing_lyc_suppress |
| | stellar_population_spectra-postprocessing_madau1995 stellar_population_spectra-postprocessing_meiksin2006 |
| | stellar_population_spectra-string_handling |
| | postprocessing_recent |

**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.Lyman_continuumSuppress.F90

| Description: | Contains a module which implements suppression of the Lyman continuum in galaxy spectra. |
| Code lines: | 55 |

**module:** stellar_population_spectra_postprocessing_lyc_suppress

| Description: | Implements suppression of the Lyman continuum in galaxy spectra. |
| Code lines: | 35 |
| Contained by: | file stellar_populations.spectra.postprocess.Lyman_continuumSuppress.F90 |

| Modules used: | iso_varying_string |
| Used by: | function stellar_population_spectrum-postprocess_index |

**subroutine:** stellar_population_spectrum_postprocess_lyc_suppress

| Description: | Suppresses all starlight in the Lyman continuum. |
| Code lines: | 10 |
18.1. Program units

**Contained by:** module `stellar_population_spectra_postprocessing_lyc_suppress`

**Modules used:** `numerical(constants)_atomic`

**subroutine:** `stellar_population_spectra_postprocess_lyc_suppress_initialize`

**Description:** Initializes the “Lyman-continuum suppression” stellar spectrum postprocessing module.

**Code lines:** 8

**Contained by:** module `stellar_population_spectra_postprocessing_lyc_suppress`

**file:** `stellar_populations.spectra.postprocess.Madau1995.F90`

**Description:** Contains a module which implements the Madau [1995] calculation of the attenuation of spectra by the intergalactic medium.

**Code lines:** 87

**module:** `stellar_population_spectra_postprocessing_madau1995`

**Description:** Implements the Madau [1995] calculation of the attenuation of spectra by the intergalactic medium.

**Code lines:** 67

**Contained by:** file `stellar_populations.spectra.postprocess.Madau1995.F90`

**Modules used:** `iso_varying_string`

**Used by:** function `stellar_population_spectrum_postprocess_index`

**subroutine:** `stellar_population_spectra_postprocess_madau1995`

**Description:** Computes the factor by which the spectrum of a galaxy at given redshift is attenuated at the given wavelength by the intervening intergalactic medium according to Madau [1995].

**Code lines:** 42

**Contained by:** module `stellar_population_spectra_postprocessing_madau1995`

**Modules used:** `numerical(constants)_atomic`

**subroutine:** `stellar_population_spectra_postprocess_madau1995_initialize`

**Description:** Initializes the “Madau1995” stellar spectrum postprocessing module.

**Code lines:** 8

**Contained by:** module `stellar_population_spectra_postprocessing_madau1995`

**file:** `stellar_populations.spectra.postprocess.Meiksin2006.F90`

**Description:** Contains a module which implements the Meiksin [2006] calculation of the attenuation of spectra by the intergalactic medium.

**Code lines:** 136

**module:** `stellar_population_spectra_postprocessing_meiksin2006`

**Description:** Implements the Meiksin [2006] calculation of the attenuation of spectra by the intergalactic medium.

**Code lines:** 116

**Contained by:** file `stellar_populations.spectra.postprocess.Meiksin2006.F90`

**Modules used:** `iso_varying_string`

**Used by:** function `stellar_population_spectrum_postprocess_index`

**subroutine:** `stellar_population_spectra_postprocess_meiksin2006`

**Description:** Computes the factor by which the spectrum of a galaxy at given redshift is attenuated at the given wavelength by the intervening intergalactic medium according to Meiksin [2006].

1235
subroutine: stellar_population_spectra_postprocess_meiksin2006_initialize
Description: Initializes the “Meiksin2006” stellar spectrum postprocessing module.
Code lines: 8
Contained by: module stellar_population_spectra_postprocessing_meiksin2006

file: stellar_populations.spectra.postprocess.identity.F90
Description: Contains a module which implements postprocessing of stellar spectra to keep only recent populations.
Code lines: 49

module: stellar_population_spectra_postprocessing_identity
Description: Implements postprocessing of stellar spectra to keep only identity populations.
Code lines: 29
Contained by: file stellar_populations.spectra.postprocess.identity.F90
Modules used: iso_varying_string
Used by: function stellar_population_spectrum_postprocess_index

subroutine: stellar_population_spectra_postprocess_identity
Description: An identity operator for postprocessing of stellar spectra (i.e. does nothing).
Code lines: 7
Contained by: module stellar_population_spectra_postprocessing_identity

subroutine: stellar_population_spectra_postprocess_identity_init
Description: Initializes the “identity” stellar spectrum postprocessing module.
Code lines: 8
Contained by: module stellar_population_spectra_postprocessing_identity

file: stellar_populations.spectra.postprocess.null.F90
Description: Contains a module which implements a null post-processing of stellar spectra.
Code lines: 51

module: stellar_population_spectra_postprocess_null
Description: Implements a null post-processing of stellar spectra.
Code lines: 31
Contained by: file stellar_populations.spectra.postprocess.null.F90

function: stellar_population_spectra_postprocess_null_get
Description: Null post-processing of stellar spectra: always returns unity.
Code lines: 7
Contained by: module stellar_population_spectra_postprocess_null

subroutine: stellar_population_spectra_postprocess_null_initialize
Description: Initializes the “Null” stellar spectrum postprocessing module.
18.1. Program units

Code lines: 9
Contained by: module `stellar_population_spectra_postprocess_null`
Modules used: `iso_varying_string`

file: `stellar_populations.spectra.postprocess.recent.F90`
Description: Contains a module which implements postprocessing of stellar spectra to keep only recent populations.
Code lines: 75

module: `stellar_population_spectra_postprocessing_recent`
Description: Implements postprocessing of stellar spectra to keep only recent populations.
Code lines: 55
Contained by: file `stellar_populations.spectra.postprocess.recent.F90`
Modules used: `iso_varying_string`
Used by: function `stellar_population_spectrum_postprocess_index`

subroutine: `stellar_population_spectra_postprocess_recent`
Description: Apply dust attenuation to galaxy spectra
Code lines: 8
Contained by: module `stellar_population_spectra_postprocessing_recent`

subroutine: `stellar_population_spectra_postprocess_recent_init`
Description: Initializes the “recent” stellar spectrum postprocessing module.
Code lines: 27
Contained by: module `stellar_population_spectra_postprocessing_recent`
Modules used: `input_parameters`

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: 178

module: `cosmological_mass_variance_filtered_power_spectrum`
Description: Implements calculation of $\sigma(M)$ via filtering of the power spectrum.
Code lines: 158
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: 9
Contained by: module `cosmological_mass_variance_filtered_power_spectrum`
Modules used: `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

1237
module cosmological_mass_variance_filtered_power_spectrum

subroutine cosmological_mass_variance_fps_state_retrieve
    Description: Retrieve the tabulation state from the file.
    Code lines: 10
end subroutine

subroutine cosmological_mass_variance_fps_state_store
    Description: Write the tabulation state to file.
    Code lines: 9
end subroutine

function variance_integral
    Description: Compute the root-variance of mass in spheres enclosing the given mass from the power spectrum.
    Code lines: 27
end function

function variance_integrand
    Description: Integrand function used in compute the variance in (real space) top-hat spheres from the power spectrum.
    Code lines: 14
end function

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
end function

file: structure_formation.critical_overdensity.F90
    Description: Contains a module which implements the critical linear theory overdensity for halo collapse.
    Code lines: 381

module critical_overdensity
    Description: Implements the critical linear theory overdensity for halo collapse.
    Code lines: 361
end module

Contained by: file structure_formation.critical_overdensity.F90
18.1. Program units

**Modules used:**
- iso_varying_string
- modules used: tables
- Used by:
  - function expansion_factor_at_formation
  - function growthrateodes
  - function dark_matter_profile_shape_gao2008
  - subroutine halo_mass_function_compute
  - subroutine merger_tree_build_do_cole2000
  - function virial_orbital_parameters_wetzel2010
  - function excursion_sets_barrier_gradient_critical_overdensity
  - function dark_matter_halo_bias_press_schechter
  - function dark_matter_halo_bias_smt
  - function halo_mass_function_sheth_tormen_differential
  - program tests_spherical-collapse_dark_energy_eds
  - program tests_spherical-collapse_dark_energy_omega_two_thirds
  - program tests_spherical-collapse_dark_energy_omega_half
  - program tests_spherical-collapse_dark_energy_omega_lambda

**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: 24
- Contained by: module critical_overdensity
- Modules used: cosmology_functions root_finder

**function:** critical_overdensity_for-collapse
- Description: Return the linear theory critical overdensity for collapse at the given cosmic time.
- Code lines: 44
- Contained by: module critical_overdensity
- Modules used: cosmology_functions galacticus_error

**function:** critical_overdensity_for-collapse_time_gradient
- Description: Return the derivative with respect to time of the linear theory critical overdensity for collapse at the given cosmic time.
- Code lines: 44
- 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
              spherical-collapse_matter_dark_energy
              spherical-collapse_matter_lambda

function: critical_overdensity_mass_scaling
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_mass_scaling_gradient
Description: Return the gradient with mass of a multiplicative, mass-dependent factor by which the
              critical overdensity should be scaled.
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_of_collapse
Description: Returns the time of collapse for a perturbation of linear theory overdensity
              criticalOverDensity.
Code lines: 29
Contained by: module critical_overdensity
Modules used: cosmology_functions

Description: Contains a module which implements calculations of critical overdensity using the fitting
              function of Kitayama and Suto [1996].
Code lines: 89

module: critical_overdensities_kitayama_suto1996
18.1. Program units

**subroutine:** critical_overdensity_kitayama_suto1996

*Description:* Tabulate the virial density contrast for the Kitayama and Suto [1996] fitting function module.

**subroutine:** critical_overdensity_kitayama_suto1996_initialize

*Description:* Initializes the $\delta_c$ calculation for the Kitayama and Suto [1996] fitting function module.

**file:** structure_formation.critical_overdensity.mass_scaling.null.F90

*Description:* Contains a module which implements a null scaling of critical overdensities for collapse.

**module:** critical_overdensity_mass_scalings_null

*Description:* Implements a null scaling of critical overdensities for collapse.

**function:** critical_overdensity_mass_scaling_gradient_null

*Description:* Returns the gradient of a mass scaling for critical overdensities that is always unity.

**function:** critical_overdensity_mass_scaling_null

*Description:* Returns a mass scaling for critical overdensities that is always unity.

**subroutine:** critical_overdensity_mass_scaling_null_initialize

*Description:* Initializes the “null” critical overdensity mass scaling method.

**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].
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: 184
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
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: 77
Contained by: module critical_overdensity_mass_scalings_wdm
Modules used: cosmological_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 set calculations.
Code lines: 335

module: excursion_sets_barriers
Description: Implements calculations of barriers for excursion set 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
18.1. Program units

function excursion_sets_first_crossing_probability_farahi
function g_1
function g_2
function g_2_integrated
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_quad
excursion_sets_barriers_remap_null
excursion_sets_barriers_remap_scale galacticus_error memory_management input_parameters 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
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
18.1. Program units

*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_quadratic

**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.

*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_remap_smt

*Used by:* subroutine excursion_sets_barrier_gradient_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:* 124

*Contained by:* module excursion_sets_barriers_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].
18. Source Code Documentation

**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, function excursion_sets_barrier_gradient, subroutine excursion_sets_barrier_gradient_initialize

**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_setsBarrier_initialize

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: inputParameters iso_varying_string

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 merging_rate
function halo_mass_function

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 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.
18. Source Code Documentation

**Code lines:** 56
**contained by:** module `excursion_sets_first_crossings`
**Modules used:**
- `excursion_sets_first_crossing_farahi`
- `excursion_sets_first_crossing_zhang_hui`
- `galacticus_error`

**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:** 654

**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:** 630
**Contained by:** file `structure_formation.excursion_sets.first_crossing_distribution.Farahi.F90`
**Modules used:**
- `fgsl`
- `iso_varying_string`
**Used by:** subroutine `excursion_sets_first_crossing_farahi_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:** 11
**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
18.1. Program units

**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:**
31

**Contained by:** module excursion_sets_first_crossing_farahi

**Modules used:**
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:**
109

**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:**
61

**Contained by:** module excursion_sets_first_crossing_farahi

**Modules used:**
umerical_interpolation

**subroutine:** excursion_sets_first_crossing_rate_tabulate_farahi

**Description:**
Tabulate the excursion set crossing rate.

**Code lines:**
155

**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:**
umerical_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:**
169
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: 148
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: 87
Contained by: module excursion_sets_first_crossing_zhang_hui
Modules used: excursion_sets_first_crossing_zhang_hui_utilities galacticus_display hui_utilities 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: subroutine excursion_sets_first_crossings_initialize 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 math_distributions_gaussian

function: g_2
Description: Returns the function \( g_2(S, S') \) in the Zhang and Hui [2006] algorithm for excursion set barrier crossing probabilities.
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 fgs1 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: 216

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: 193
Contained by: file structure_formation.excursion_sets.first_crossing_distribution.Zhang_Hui_high_order.F90
Modules used: fgs1
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:* 132
*Contained by:* module excursion_sets_first_crossing_zhang_hui_high
*Modules used:*
- excursion_sets_first_crossing_zhang_hui_high
- galacticus_display
- hui_utilities
- 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_crossing_linear_barrier_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**
18.1. Program units

**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

**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

**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

**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
Source Code Documentation

**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`
- `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`

**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`

**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`
18.1. Program units

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: 95

module: dark_matter_halo_biases_tinker2010
Description: Implements calculations of halo bias using the fitting function of Tinker et al. [2010].
Code lines: 75
Contained by: file structure_formation.halo_bias.Tinker2010.F90
Used by: subroutine dark_matter_halo_bias_.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: 34
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**: 187

**module**: halo_mass_function
**Code lines**: 169
**Contained by**: file structure_formation.halo_mass_function.F90
**Modules used**: f gsl iso_varying_string
**Used by**: program tests_excursion_sets program optimal_sampling_smf subroutine merger_tree_build_initialize subroutine merger_tree_construct_mass_{
function merger_tree_construct_mass_{
function sampling_halo_mf function sampling_stellar_mf program tests_halo_mass_function_{
tinker

**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**: 21
**Contained by**: module halo_mass_function
**Modules used**: cosmological_parameters numerical_integration

**function**: halo_mass_function_differential
**Description**: Return the differential halo mass function for mass [M☉] at time.
**Code lines**: 11
**Contained by**: module halo_mass_function

**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 \texttt{massLow} and \texttt{massHigh}.
*Code lines:* 18
*Contained by:* module \texttt{halo_mass_function}
*Modules used:* \texttt{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:* 60

**module: halo_mass_function_press_schechter**
*Description:* Implements generation of a tabulated power-law primordial power spectrum.
*Code lines:* 40
*Contained by:* file \texttt{structure_formation.halo_mass_function.Press-Schechter.F90}
*Used by:* subroutine \texttt{halo_mass_function_-initialize}

**function: halo_mass_function_differential_press_schechter**
*Description:* Compute the Press-Schechter halo mass function.
*Code lines:* 13
*Contained by:* module \texttt{halo_mass_function_press_schechter}
*Modules used:* \texttt{cosmological_parameters} \texttt{excursion_sets_first_crossings} \texttt{power_spectra}

**subroutine: halo_mass_function_press_schechter_initialize**
*Description:* Initializes the “Press-Schechter mass function” module.
*Code lines:* 9
*Contained by:* module \texttt{halo_mass_function_press_schechter}
*Modules used:* \texttt{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:* 64

**module: halo_mass_function_sheth_tormen**
*Description:* Implements generation of a tabulated power-law primordial power spectrum.
*Code lines:* 44
*Contained by:* file \texttt{structure_formation.halo_mass_function.Sheth-Tormen.F90}
*Used by:* subroutine \texttt{halo_mass_function_-initialize}

**function: halo_mass_function_sheth_tormen_differential**
*Description:* Compute the Sheth-Tormen halo mass function.
*Code lines:* 17
*Contained by:* module \texttt{halo_mass_function_sheth_tormen}
*Modules used:* \texttt{cosmological_parameters} \texttt{critical_overdensity} \texttt{numerical_constants_math} \texttt{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


Description:  Contains a module which generates a tabulated Tinker2008 halo mass function.

Code lines:  124

module:  halo_mass_function_tinker2008

Description:  Implements generation of a tabulated power-law primordial power spectrum.

Code lines:  104

Contained by:  file  structure_formation.halo_mass_function.Tinker2008.F90

Used by:  subroutine  halo_mass_function_tinker2008_initialize

function:  halo_mass_function_differential_tinker2008

Description:  Compute the Tinker et al. [2008] halo mass function.

Code lines:  49

Contained by:  module  halo_mass_function_tinker2008

Modules used:  cosmological_parameters  cosmology_functions
fgsl  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.linear_growth.F90

Description:  Contains a module which implements linear growth factor calculations.

Code lines:  328

module:  linear_growth

Description:  Implements the virial overdensity for halos.

Code lines:  308

Contained by:  file  structure_formation.linear_growth.F90

Modules used:  fgsl  iso_varying_string

Used by:  function  dark_matter_profile_concentration_prada2011
subroutine  galacticus_growth_factor_output
subroutine  galacticus_state_retrieve
subroutine  critical_overdensity_kitayama_suto1996
subroutine  halo_mass_function_compute
function  halo_mass_function_differential_tinker2008
function power_spectrum_nonlinear_peacockdodds1996
function power_spectrum_nonlinear_linear

subroutine make_table

program tests_linear_growth_eds
program tests_linear_growth_- cosmological_constant
program tests_linear_growth_- energy
program tests_linear_growth_dark_- energy
program tests_linear_growth_dark_- energy_omega_half
program tests_linear_growth_dark_- energy_omega_two_thirds
program tests_linear_growth_dark_- energy_lambda
program tests_linear_growth_dark_- energy_open

subroutine: interpolate_in_wavenumber

  Description: Find interpolating factors in the wavenumber dimension for linear growth factor calculations.
  Code lines: 18
  Contained by: module linear_growth
  Modules used: numerical_interpolation

function: linear_growth_factor

  Description: Return the linear growth factor.
  Code lines: 94
  Contained by: module linear_growth
  Modules used: cosmology_functions galacticus_error 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: 74
  Contained by: module linear_growth
  Modules used: cosmology_functions galacticus_error 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

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_formroduction.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:** 182
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: 161
Contained by: file structure_formation.linear_growth.simple.F90
Modules used: cosmological_parameters cosmology_functions
fgal 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: 9
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 $a_{\text{Expansion}}$, normalized such that $D(1) = 1$ for a simple matter plus cosmological constant cosmology.
Code lines: 79
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: 284

module: power_spectra
Description: Implements the cosmological power spectrum.
Code lines: 264
Contained by: file structure_formation.power_spectrum.F90
18. Source Code Documentation

---

**Modules used:** iso_varying_string  
**Used by:**  
program tests_excursion_sets  
function dark_matter_halo_mass_accretion_time_zhao2009  
function dark_matter_profile_concentration_nfw1996  
function dark_matter_profile_shape_gao2008  
subroutine halo_mass_function_compute  
module modified_press_schechter_branching  
subroutine merger_tree_write  
function collapsing_mass_root  
function excursion_sets_barrier_gradient_critical_overdensity  
function dark_matter_halo_bias_press_schechter  
function dark_matter_halo_bias_smt  
function dark_matter_halo_bias_tinker2010  
function halo_mass_function_differential_press_schechter  
function halo_mass_function_differential_tinker2008  
function power_spectrum_nonlinear_cosmicemu  
function power_spectrum_nonlinear_peacockdodds1996  
program tests_power_spectrum  

**function:** cosmological_mass_root_variance  
**Description:** Computes the fractional mass fluctuation in real-space spherical top hats enclosing mass $M$.  
**Code lines:** 13  
**Contained by:** module power_spectra

**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 $M$.  
**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 $M$ 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 $\log M$. The default normalization, $\sigma_9 = 0.807$, is taken from Komatsu et al. [2010].  
**Code lines:** 81  
**Contained by:** module power_spectra
18.1. Program units

**Modules used:**
- cosmological_mass_variance_filtered-
power_spectrum
galacticus_error
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 \( k = \text{wavenumber} \ [\text{Mpc}^{-1}] \).

*Code lines:* 32

*Contained by:* module power_spectra

**Modules used:**
- cosmological_parameters
galacticus_error
numerical_constants_math
primordial_power_spectra_transferred

**function:** power_spectrum_dimensionless

*Description:* Return the dimensionless power spectrum, \( \Delta^2(k) \), for \( k = \text{wavenumber} \ [\text{Mpc}^{-1}] \).

*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, \( \frac{d \ln P(k)}{d \ln k} \), for \( k = \text{wavenumber} \ [\text{Mpc}^{-1}] \).

*Code lines:* 9

*Contained by:* module power_spectra

*Modules used:* primordial_power_spectra
transfer_functions

**function:** sigma_8

*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:* 158

**module:** power_spectra_nonlinear_cosmicemu

*Description:* Implements the nonlinear power spectrum using the code of Lawrence et al. [2010].

*Code lines:* 138

*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:* 103

*Contained by:* module `power_spectra_nonlinear_cosmicemu`

*Modules used:*
- `cosmological_parameters`
- `cosmology_functions`
- `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`

**function: power_spectrum_nonlinear**

*Description:* Return the nonlinear power spectrum for \( k = \text{wavenumber} \text{[Mpc}^{-1}] \) at the given cosmic time \( \text{[Gyr]} \).

*Code lines:* 12

*Contained by:* module `power_spectra_nonlinear`

**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`
- `peacockdodds1996`

**file: structure_formation.power_spectrum.nonlinear.PeacockDodds1996.F90**

*Description:* Contains a module which implements the nonlinear power spectrum using the algorithm of Peacock and Dodds [1996].

*Code lines:* 97
module: power_spectra_nonlinear_peacockdodds1996
Description: Implements the nonlinear power spectrum using the algorithm of Peacock and Dodds [1996].
Code lines: 77
Contained by: file structure_formation.power_spectrum.nonlinear.PeacockDodds1996.F90
Used by: subroutine power_spectrum_nonlinear_peacockdodds1996

function: power_spectrum_nonlinear_peacockdodds1996
Description: Return a nonlinear power spectrum equal using the algorithm of Peacock and Dodds [1996].
Code lines: 53
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_linear

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_cosmicemu 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_parameters
                primordial_power_spectrum_power_law
function: primordial_power_spectrum
  Description: Return the CDM primordial power spectrum for \( k = \text{wavenumber} \) [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 \( k = \text{wavenumber} \) [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
  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 Komatsu et al. [2010].
  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: fgs1

subroutine: primordial_power_spectrum_power_law_state_store
Description: Write the tablulation state to file.
Code lines: 9
Contained by: module primordial_power_spectrum_power_law
Modules used: fgs1

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
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: 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: 108

module: power_spectrum_window_functions_sharp_kspace

Description: Implements a sharp in $k$-space window function for power spectrum variance computation.

Code lines: 87

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: 41
Contained by: module power_spectrum_window_functions_sharp_kspace
Modules used: cosmological_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: 83

module: power_spectrum_window_functions_top_hat
Description: Implements top-hat window function for power spectrum variance computation.
Code lines: 62
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: 23
Contained by: module power_spectrum_window_functions_top_hat
Modules used: cosmological_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: 155
18.1. Program units

module: power_spectrum_window_functions_th_kss_hybrid
Description: Implements top-hat window function for power spectrum variance computation.
Code lines: 134
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 \( a = \text{[powerSpectrumWindowFunctionSharpKSpaceNormalization]} \). 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 \( \beta = \text{[powerSpectrumWindowFunctionSharpKSpaceTopHatRadiiRatio]} \).
Code lines: 46
Contained by: module power_spectrum_window_functions_th_kss_hybrid
Modules used: cosmological_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 \( a = \text{[powerSpectrumWindowFunctionSharpKSpaceNormalization]} \). 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 \( \beta = \text{[powerSpectrumWindowFunctionSharpKSpaceTopHatRadiiRatio]} \).
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: 52
Contained by: module power_spectrum_window_functions_th_kss_hybrid
Modules used: cosmological_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: 349

module: spherical-collapse.matter.dark.energy
Description: Implements calculations of spherical top hat collapse in cosmologies containing matter and dark energy.
Code lines: 329
Contained by: file structure_formation.spherical-collapse.matter.dark.energy.F90
Modules used: fgs1 iso_c_binding iso_varying_string
18. Source Code Documentation

**function**: expansionrateperturbation

*Description:* Return the expansion rate of a spherical top-hat perturbation in a dark energy universe given an initial perturbation amplitude $\epsilon_{\text{Perturbation}}$.

*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:* 115

*Contained by:* module spherical_collapse_matter_dark_energy

*Modules used:* cosmology_functions, galacticus_display, galacticus_error, 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 $\epsilon_{\text{Perturbation}}$.

*Code lines:* 46

*Contained by:* module spherical_collapse_matter_dark_energy

*Modules used:* cosmology_functions, fodeiv2, odeiv2_solver

**function**: perturbationodes

*Code lines:* 20

*Contained by:* module spherical_collapse_matter_dark_energy

*Modules used:* cosmology_functions

**function**: radiusperturbation

*Description:* Return the radius of a spherical top-hat perturbation in a dark energy universe given an initial perturbation amplitude $\epsilon_{\text{Perturbation}}$.

*Code lines:* 8

*Contained by:* module spherical_collapse_matter_dark_energy

*Modules used:* cosmology_functions

**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_delta_virial_initialize

*Code lines:* 1272

---

*Used by:* subroutine galacticus_state_retrieve, subroutine galacticus_state_store, subroutine critical_overdensity_initialize, subroutine virial_density_contrast_initialize

---

1272
18.1. Program units

**Description:** Initializes the $\Delta_{\text{vir}}$ calculation for the spherical collapse module.

**Code lines:** 24

**Contained by:** module `sphericalCollapse_matter_dark_energy`

**Modules used:**
- `input_parameters`

**subroutine:** `sphericalCollapse_dark_energy_virial_density_contrast`

**Description:** Tabulate the virial density contrast for the spherical collapse model.

**Code lines:** 11

**Contained by:** module `sphericalCollapse_matter_dark_energy`

**Modules used:**
- `tables`

**subroutine:** `sphericalCollapse_matter_dark_energy_state_retrieve`

**Description:** Retrieve the tabulation state from the file.

**Code lines:** 8

**Contained by:** module `sphericalCollapse_matter_dark_energy`

**subroutine:** `sphericalCollapse_matter_dark_energy_state_store`

**Description:** Write the tabulation state to file.

**Code lines:** 8

**Contained by:** module `sphericalCollapse_matter_dark_energy`

**file:** `structureFormation.sphericalCollapse.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:** 321

**module:** `sphericalCollapse_matter_lambda`

**Code lines:** 300

**Contained by:** file `structureFormation.sphericalCollapse.matter.lambda.F90`

**Modules used:**
- `fgsl`
- `iso_c_binding`

**Used by:**
- subroutine `galacticus_state_retrieve`
- subroutine `galacticus_state_store`
- subroutine `critical_overdensity_initialize`
- subroutine `virial_density_contrast_initialize`

**function:** `amaximumroot`

**Description:** Root function for maximum expansion radius.

**Code lines:** 6

**Contained by:** module `sphericalCollapse_matter_lambda`

**function:** `collapseroot`

**Code lines:** 5

**Contained by:** module `sphericalCollapse_matter_lambda`

**subroutine:** `make_table`

**Description:** Tabulate $\delta_{\text{crit}}$ or $\Delta_{\text{vir}}$ vs. time.

**Code lines:** 97

**Contained by:** module `sphericalCollapse_matter_lambda`

**Modules used:**
- `cosmology_functions`
- `galacticus_error`
- `kind_numbers`
- `linear_growth`
18. Source Code Documentation

**function:** perturbation_integrand
- **Code lines:** 15
- **Contained by:** module `sphericalCollapse_matter_lambda`

**function:** perturbation_maximum_radius
- **Description:** Find the maximum radius of a perturbation with initial curvature $\epsilon_{\text{Perturbation}}$.
- **Code lines:** 37
- **Contained by:** module `sphericalCollapse_matter_lambda`
- **Modules used:** `root_finder`

**subroutine:** sphericalCollapse_delta_critical_initialize
- **Description:** Initializes the $\delta_{\text{crit}}$ calculation for the spherical collapse module.
- **Code lines:** 9
- **Contained by:** module `sphericalCollapse_matter_lambda`
- **Modules used:** `iso_varying_string`

**subroutine:** sphericalCollapse_delta_virial_initialize
- **Description:** Initializes the $\Delta_{\text{vir}}$ calculation for the spherical collapse module.
- **Code lines:** 9
- **Contained by:** module `sphericalCollapse_matter_lambda`
- **Modules used:** `iso_varying_string`

**subroutine:** sphericalCollapse_critical_overdensity
- **Description:** Tabulate the critical overdensity for collapse for the spherical collapse model.
- **Code lines:** 12
- **Contained by:** module `sphericalCollapse_matter_lambda`
- **Modules used:** `tables`

**subroutine:** sphericalCollapse_matter_lambda_state_retrieve
- **Description:** Retrieve the tabulation state from the file.
- **Code lines:** 8
- **Contained by:** module `sphericalCollapse_matter_lambda`

**subroutine:** sphericalCollapse_matter_lambda_state_store
- **Description:** Write the tabulation state to file.
- **Code lines:** 8
- **Contained by:** module `sphericalCollapse_matter_lambda`

**subroutine:** sphericalCollapse_virial_density_contrast
- **Description:** Tabulate the virial density contrast for the spherical collapse model.
- **Code lines:** 11
- **Contained by:** module `sphericalCollapse_matter_lambda`
- **Modules used:** `tables`

**function:** tcollapse
- **Code lines:** 28
- **Contained by:** module `sphericalCollapse_matter_lambda`
18.1. Program units

**Modules used:** numerical_integration

**file:** structure_formation.transfer_function.BBKS.F90
**Description:** Contains a module which generates a tabulated transfer function using the BBKS fitting formula.
**Code lines:** 133

**module:** transfer_function_bbks
**Description:** Implements generation of a tabulated transfer function using the BBKS fitting formula.
**Code lines:** 113
**Contained by:** file structure_formation.transfer_function.BBKS.F90
**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:** 25
**Contained by:** module transfer_function_bbks
**Modules used:** iso_varying_string

**subroutine:** transfer_function_bbks_make
**Description:** Build a transfer function using the BBKS fitting formula.
**Code lines:** 35
**Contained by:** module transfer_function_bbks
**Modules used:** cosmological_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

**file:** structure_formation.transfer_function.CMBFast.F90
**Description:** Contains a module which generates a tabulated transfer function using CMBFast.
**Code lines:** 126

**module:** transfer_function_cmbfast
**Description:** Implements generation of a tabulated transfer function using CMBFast.
**Code lines:** 106
**Contained by:** file structure_formation.transfer_function.CMBFast.F90
**Modules used:** iso_varying_string
18. Source Code Documentation

**Used by:** subroutine `transfer_function_initialize`

**subroutine:** `transfer_functioncmbfast_initialize`
- **Description:** Initializes the “transfer function from CMBFast” module.
- **Code lines:** 8
- **Contained by:** module `transfer_functioncmbfast`

**subroutine:** `transfer_functioncmbfast_make`
- **Description:** Build a transfer function using CMBFAST.
- **Code lines:** 73
- **Contained by:** module `transfer_functioncmbfast`
- **Modules used:**
  - `cosmological_parameters`
  - `fox_wxml`
  - `galacticus_input_paths`
  - `input_parameters`
  - `numerical_constants_astronomical`
  - `string_handling`
  - `system_command`
  - `transfer_functions_file`

**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:** 241

**module:** `transfer_functioneisenstein_hu`
- **Description:** Implements generation of a tabulated transfer function using the Eisenstein & Hu fitting formula.
- **Code lines:** 221
- **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_functioncmbfast_initialize`

**subroutine:** `transfer_functioneisenstein_hu_initialize`
- **Description:** Initializes the “transfer function from Eisenstein & Hu” module.
- **Code lines:** 77
- **Contained by:** module `transfer_functioneisenstein_hu`
- **Modules used:**
  - `input_parameters`

**subroutine:** `transfer_functioneisenstein_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 [transferFunctionWdmCutOffScale].
- **Code lines:** 85
- **Contained by:** module `transfer_functioneisenstein_hu`
- **Modules used:**
  - `cosmological_parameters`
  - `memory_management`
  - `numerical_constants_math`
  - `numerical_ranges`

**subroutine:** `transfer_functioneisenstein_hu_state_retrieve`
- **Description:** Retrieve the tabulation state from the file.
- **Code lines:** 10
- **Contained by:** module `transfer_functioneisenstein_hu`
18.1. Program units

**Modules used:** fgs1

**subroutine:** transfer_function_eisenstein_hu_state_store
- **Description:** Write the tabulation state to file.
- **Code lines:** 9
- **Contained by:** module transfer_function_eisenstein_hu
- **Modules used:** fgs1

**file:** structure_formation.transfer_function.F90
- **Code lines:** 193

**module:** transfer_functions
- **Code lines:** 175
- **Contained by:** file structure_formation.transfer_function.F90
- **Modules used:** fgs1 iso_varying_string
- **Used by:** subroutine galacticus_state_retrieve function power_spectrum_logarithmic_derivative
  - function primordial_power_spectrum_transferred

**function:** transfer_function
- **Description:** Return the transfer function for \( k = \text{wavenumber} \) [Mpc\(^{-1}\)].
- **Code lines:** 30
- **Contained by:** module transfer_functions
- **Modules used:** numerical_interpolation

**subroutine:** transfer_function_initialize
- **Description:** Initializes the transfer function module.
- **Code lines:** 60
- **Contained by:** module transfer_functions
- **Modules used:** galacticus_state Retrieve input_parameters
  - transfer_function_bbks transfer_function_cmbfast
  - 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 \( k = \text{wavenumber} \) [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
18. Source Code Documentation

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:** 242

**module: transfer_functions_file**

- **Description:** Implements reading of a tabulated transfer function from a file.
- **Code lines:** 222
- **Contained by:** file `structure_formation.transfer_function.file.F90`
- **Modules used:** `iso_varying_string`
- **Used by:** subroutine `transfer_function_cmbfast_-` subroutine `transfer_function_initialize`

**subroutine: transfer_function_file_initialize**

- **Description:** Initializes the “transfer function from file” module.
- **Code lines:** 21
- **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:** 144
- **Contained by:** module `transfer_functions_file`
- **Modules used:** `cosmological_parameters` `fox_dom` `galacticus_display` `galacticus_error` `io_xml` `memory_management` `numerical_comparison` `numerical_constants_math` `numerical_ranges`

**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:** 74

**module: transfer_function_null**
18.1. Program units

Description: Implements a null transfer function.
Code lines: 54
Contained by: file structure_formation.transfer_function.null.F90
Used by: subroutine transfer_function_initialize

**subroutine: transfer_function_null_initialize**

Description: Initializes the “null transfer function” module.
Code lines: 9
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**

Description: Contains a module which implements calculations of virial overdensity using the fitting function of Bryan and Norman [1998].
Code lines: 103

**module: virial_densities_bryan_norman**

Code lines: 82
Contained by: file structure_formation.virial_density_contrast.Bryan_Norman.F90
Used by: subroutine virial_density_contrast_.-initialize

**subroutine: virial_density_bryan_norman**

Description: Tabulate the virial density contrast for the Bryan and Norman [1998] fitting function module.
Code lines: 38
Contained by: module virial_densities_bryan_norman
Modules used: cosmology_functions numerical_constants_math tables

**subroutine: virial_density_bryan_norman_initialize**

Description: Initializes the $\Delta_{\text{vir}}$ calculation for the Bryan and Norman [1998] fitting function module.
Code lines: 22
Contained by: module virial_densities_bryan_norman
Modules used: cosmological_parameters galacticus_error iso_varying_string numerical_comparison

**file: structure_formation.virial_density_contrast.F90**

Description: Contains a module which implements the virial overdensity for halos.
Code lines: 226

**module: virial_density_contrast**

Description: Implements the virial overdensity for halos.
function: halo_virial_density_contrast
Description: Return the halo virial overdensity.
Code lines: 36
Contained by: module virial_density_contrast
Modules used: cosmology_functions galacticus_error

function: halo_virial_density_contrast_rate_of_change
Description: Return the halo virial overdensity rate of change.
Code lines: 36
Contained by: module virial_density_contrast
Modules used: cosmology_functions galacticus_error

subroutine: virial_density_contrast_initialize
Description: Initializes the virial overdensity module.
Code lines: 62
Contained by: module virial_density_contrast
galacticus_error input_parameters
sphericalCollapseMatter_dark_energy sphericalCollapseMatter_lambda
virialDensities_bryan_norman virialDensities_fixed
virialDensities_kitayama_suto1996

subroutine: virial_density_contrast_retabulate
Description: Recompute the look-up tables for virial density contrast.
Code lines: 20
Contained by: module virial_density_contrast

subroutine: virial_density_contrast_state_retrieve
Description: Reset the tabulation if state is to be retrieved. This will force tables to be rebuilt.
Code lines: 9
18.1. Program units

Contained by: module virial_density_contrast
Modules used: fgsl

Description: Contains a module which implements calculations of virial overdensity using the fitting function of Kitayama and Suto [1996].
Code lines: 89

module: virial_densities_kitayama_suto1996
Code lines: 68
Used by: subroutine virial_density_contrast_-_initialize

subroutine: virial_density_kitayama_suto1996
Description: Tabulate the virial density contrast for the Kitayama and Suto [1996] fitting function module.
Code lines: 34
Contained by: module virial_densities_kitayama_suto1996
Modules used: cosmology_functions numerical_constants_math tables

subroutine: virial_density_kitayama_suto1996_initialize
Description: Initializes the \( \Delta_{\text{vir}} \) calculation for the Kitayama and Suto [1996] fitting function module.
Code lines: 16
Contained by: module virial_densities_kitayama_suto1996
Modules used: cosmological_parameters galacticus_error iso_varying_string numerical_comparison

file: structure_formation.virial_density_contrast.fixed.F90
Description: Contains a module which implements calculations of virial overdensity using a fixed value.
Code lines: 130

module: virial_densities_fixed
Description: Implements calculations of virial overdensity using a fixed value.
Code lines: 110
Contained by: file structure_formation.virial_density_contrast.fixed.F90
Used by: subroutine virial_density_contrast_-_initialize

subroutine: virial_density_fixed
Description: Tabulate the virial density contrast assuming a fixed value.
Code lines: 37
Contained by: module virial_densities_fixed
Modules used: cosmology_functions tables

subroutine: virial_density_fixed_initialize
Description: Initializes the \( \Delta_{\text{vir}} \) calculation for the fixed value implementation.
Code lines: 46
Contained by: module virial_densities_fixed
18. Source Code Documentation

**Modules used:**
- galacticus_error
- input_parameters
- iso_varying_string

**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:* subroutine chemical_state_atomic_cie_cloudy_create
- subroutine cooling_function_atomic_cie_cloudy_create
- subroutine intergalactic_medium_state_refast_initialize
- subroutine stellar_population_spectra_conroy_initialize_imf
- function power_spectrum_nonlinear_cosmicemu
- subroutine transfer_function_cmbfast_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

*Contained by:* 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
18.1. Program units

**Description:** Tests the HDF5 I/O module.

**Code lines:** 664

**Contained by:** file tests.IO.HDF5.F90

**Modules used:**
- hdf5
- io_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:**
- fox_dom
- io_xml
- 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:** 89

**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:** 67

**Contained by:** file tests.NFW96_concentration.dark_energy.F90

**Modules used:**
- cosmological_parameters
- cosmology_functions
- dark_matter_profiles_concentrations
- 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

**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: fgsl 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: 82

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: 62
Contained by: file tests.Prada2011_concentration.F90
Modules used: cosmological_parameters cosmology_functions
dark_matter_profiles_concentrations galacticus_nodes
input_parameters iso_varying_string
memory_management 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: 127

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: 106
Contained by: file tests.Zhao2009_algorithms.EdS.F90
Modules used: cosmology_functions
dark_matter_profiles_concentrations
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: 127

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: 106
Contained by: file tests.Zhao2009_algorithms.dark_energy.F90

Modules used:
- cosmology_functions
- dark_matter_profiles_concentrations
- dark_matter_halo_mass_accretion_histories
- galacticus_input_paths
- galacticus_nodes
- input_parameters
- iso_varying_string
- memory_management
- string_handling
- unit_tests

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: 127

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: 106
Contained by: file tests.Zhao2009_algorithms.open.F90

Modules used:
- cosmology_functions
- dark_matter_profiles_concentrations
- dark_matter_halo_mass_accretion_histories
- galacticus_input_paths
- galacticus_nodes
- input_parameters
- iso_varying_string
- 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
18.1. Program units

file: tests.arrays.F90
    Description: Contains a program to test the array functions.
    Code lines: 180

program: test_array_monotonicity
    Description: Tests that array functions.
    Code lines: 160
    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.EdS.F90
    Code lines: 66

program: tests_comoving_distance_eds
    Description: Tests comoving distance calculations for an Einstein-de Sitter Universe. Distances calculated using Python implementation of Ned Wright’s cosmology calculator available from: http://www.astro.ucla.edu/~wright/CC.python
    Code lines: 48
    Contained by: file tests.comoving_distance.EdS.F90
    Modules used: cosmology_functions cosmology_functions_options input_parameters iso_varying_string memory_management unit_tests

file: tests.comoving_distance.dark_energy.F90
18. Source Code Documentation

Code lines: 66

program: tests_comoving_distance_dark_energy
Description: Tests comoving distance calculations for a dark energy Universe. Distances calculated using Python implementation of Ned Wright's cosmology calculator available from: http://www.astro.ucla.edu/wright/CC.python
Code lines: 48
Contained by: file tests.comoving_distance.dark_energy.F90
Modules used: cosmology_functions cosmology_functions_options
input_parameters iso_varying_string
memory_management unit_tests

file: tests.comoving_distance.open.F90
Code lines: 66

program: tests_comoving_distance_open
Description: Tests comoving distance calculations for an open Universe. Distances calculated using Python implementation of Ned Wright's cosmology calculator available from: http://www.astro.ucla.edu/wright/CC.python
Code lines: 48
Contained by: file tests.comoving_distance.open.F90
Modules used: cosmology_functions cosmology_functions_options
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.EdS.F90
Code lines: 55

program: tests_cosmic_age_eds
Description: Tests cosmic age calculations for an Einstein-de Sitter Universe. Ages calculated using Python implementation of Ned Wright's cosmology calculator available from: http://www.astro.ucla.edu/wright/CC.python
Code lines: 37
Contained by: file tests.cosmic_age.EdS.F90
Modules used: cosmology_functions input_parameters
iso_varying_string memory_management
unit_tests

file: tests.cosmic_age.cosmological_constant.F90
18.1. Program units

Code lines: 55

Program: tests_cosmic_age_cosmological_constant

Description: Tests cosmic age calculations for a cosmological constant Universe. Ages calculated using Python implementation of Ned Wright’s cosmology calculator available from: http://www.astro.ucla.edu/~wright/CC.python

Code lines: 37

Contained by: file tests.cosmic_age.cosmological_constant.F90

Modules used: cosmology_functions  input_parameters
              iso_varying_string  memory_management
              unit_tests

File: tests.cosmic_age.dark_energy.closed.F90

Code lines: 67

Program: tests_cosmic_age_dark_energy_closed

Description: Tests cosmic age calculations for a dark energy Universe with no dark energy but more than the critical density of matter such that it recollapses. Ages calculated using Python implementation of Ned Wright’s cosmology calculator available from: http://www.astro.ucla.edu/~wright/CC.python

Code lines: 49

Contained by: file tests.cosmic_age.dark_energy.closed.F90

Modules used: cosmological_parameters  cosmology_functions
              input_parameters  iso_varying_string
              memory_management  numerical_constants_math
              unit_tests

File: tests.cosmic_age.dark_energy.cosmological_constant.F90

Code lines: 55

Program: tests_cosmic_age_dark_energy_cosmological_constant

Description: Tests cosmic age calculations for a dark energy Universe where dark energy acts as a cosmological constant. Ages calculated using Python implementation of Ned Wright’s cosmology calculator available from: http://www.astro.ucla.edu/~wright/CC.python

Code lines: 37

Contained by: file tests.cosmic_age.dark_energy.cosmological_constant.F90

Modules used: cosmology_functions  input_parameters
              iso_varying_string  memory_management
              unit_tests

File: tests.cosmic_age.dark_energy.omegaMinusOneThird.F90

Code lines: 56

Program: tests_cosmic_age_dark_energy_omega_minus_one_third

File: tests.cosmic_age.dark_energy.omegaMinusOneThird.F90

Code lines: 55
**Description:** Tests cosmic age calculations for a dark energy Universe where dark energy has equation of state \( w = -1/3 \), in which case the expansion history should be the same as for a model with no dark energy. Ages calculated using Python implementation of Ned Wright’s cosmology calculator available from: http://www.astro.ucla.edu/~wright/CC.python

**Code lines:** 38

**Contained by:** file `tests.cosmic_age.dark_energy.omegaMinusOneThird.F90`

**Modules used:**
- `cosmology_functions`
- `input_parameters`
- `iso_varying_string`
- `memory_management`
18.1. Program units

unit_tests

file: tests.cosmic_age.open.F90
Code lines: 55

program: tests.cosmic_age_open
Description: Tests cosmic age calculations for an open Universe. Ages calculated using Python implementation of Ned Wright’s cosmology calculator available from: http://www.astro.ucla.edu/~wright/CC.python
Code lines: 37
Contained by: file tests.cosmic_age.open.F90
Modules used: cosmology_functions
iso_varying_string
input_parameters
memory_management
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: 87

program: tests_halo_mass_function_tinker
Description: Tests the Tinker et al. [2008] mass function by comparing to Jeremy Tinker’s code.
Code lines: 66
Contained by: file tests.halo_mass_function.Tinker.F90
Modules used: cosmological_parameters
critical_overdensity
cosmology_functions
cosmology_functions
file_utilities
input_parameters
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
18. Source Code Documentation

*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*
*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:* fgsl iso_c_binding
*Used by:* program test_integration
18.1. Program units

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

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: fgsl numerical_interpolation unit_tests

file: tests.kepler_orbits.F90
Code lines:  138

program: tests_kepler_orbits
Description:  Tests for orbital parameter conversions.
Code lines:  120
18. Source Code Documentation

**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:** 54

**program:** tests_linear_growth_eds

**Description:** Tests linear growth calculations.

**Code lines:** 36

**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:** 56

**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:** 38

**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:** 56

**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:** 38

**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

**Code lines:** 52

**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:** 34

**Contained by:** file tests.linear_growth.open.F90
18.1. Program units

Modules used: cosmology_functions input_parameters
iso_varying_string linear_growth
unit_tests

file: tests.make_ranges.F90
Description: Contains a program to test the numerical range making code.
Code lines: 52

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: 98

program: test_mass_distributions
Description: Tests mass distributions.
Code lines: 78
Contained by: file tests.mass_distributions.F90
Modules used: coordinates mass_distributions
memory_management unit_tests

file: tests.math_special_functions.F90
Description: Contains a program to test mathematical special functions.
Code lines: 92

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
18. Source Code Documentation

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:
galacticus_error
array_utilities
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:
galacticus_nodes
galactic_structure_options

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: 60

program: tests_power_spectrum
Description: Tests power spectrum calculations.
18.1. Program units

Code lines: 40
Contained by: file tests.power_spectrum.F90
Modules used: cosmological_parameters
iso_varying_string
numerical_constants_math
unit_tests
input_parameters
memory_management
power_spectra

file: tests.root_finding.F90
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
test_root_finding_functions
unit_tests

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.
18. Source Code Documentation

**Code lines:** 45  
**Modules used:** arrays_search, kind_numbers

---

**file:** tests.search.F90  
**Code lines:** 45  
**Modules used:** iso_varying_string, unit_tests

---

**file:** tests.sigma.F90  
**Code lines:** 69

**program:** tests_sigma  
**Description:** Tests

---

**file:** tests.sort.F90  
**Code lines:** 45

**program:** test_sort  
**Description:** Tests of sorting functions.

---

**file:** tests.spherical_collapse.dark_energy.EdS.F90  
**Code lines:** 66  
**Modules used:** cosmological_parameters, input_parameters, iso_varying_string, memory_management, numerical_constants_math, numerical_ranges, power_spectra, unit_tests

---

**file:** tests.spherical_collapse.dark_energy.Omega_constant_minus_0.6.F90  
**Code lines:** 61

**program:** tests_spherical_collapse_dark_energy_omega_zero_point_six

---

**file:** tests.spherical_collapse.dark_energy.constantEoSminus0.6.F90  
**Code lines:** 61  
**Modules used:** cosmology_functions, critical_overdensity, input_parameters, iso_varying_string, memory_management, numerical_constants_math, unit_tests, virial_density_contrast

---

**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 DATA Thief.

Code lines: 40
18. Source Code Documentation

Contained by:  file tests.spherical_collapse.dark_energy.constantEoSminus0.6.F90
Modules used:  
             cosmolology_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:  61

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:  40
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:  64

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:  43
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:  65

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:  44
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
18.1. Program units

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: 69

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: 48
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: 63

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: 42
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: 61

program: tests.spherical_collapse.flat
Description: Tests spherical collapse calculations for a flat Universe. Compares results to the fitting formula of Bryan and Norman [1998].
Code lines: 41
Contained by: file tests.spherical_collapse.flat.F90
Modules used: cosmology_functions input_parameters
                iso_varying_string memory_management
                numerical_constants_math unit_tests
                virial_density_contrast

file: tests.spherical_collapse.open.F90
Description: Contains a program which tests spherical collapse calculations for an open Universe.
Code lines: 61

**program: tests_sphericalCollapse_open**

Description: Tests spherical collapse calculations for an open Universe. Compares results to the fitting formula of Bryan and Norman [1998].
Code lines: 41
Contained by: file tests.spherical_collapse.open.F90
Modules used: cosmology_functions input_parameters
iso_varying_string memory_management
numerical_constants_math unit_tests
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 unit_tests

**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 unit_tests

**file: tests.tree_branch_destroy.F90**

Code lines: 78

**program: tests_tree_branch_destroy**

Code lines: 60
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**
18.1. Program units

**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`, function `bondi_hoyle_lyttleton_accretion_rate`, subroutine `node_component_black_hole_standard_mass_accretion_rate`

**function:** `ideal_gas_jeans_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

**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:** `utility.IO.HDF5.F90`
Description: Contains a module that implements simple and convenient interfaces to a variety of HDF5 functionality.

Code lines: 10537

**Module: io_hdf5**

Description: Implements simple and convenient interfaces to a variety of HDF5 functionality.

Code lines: 10514

Contained by: file utility.IO.HDF5.F90

Modules used: hdf5

iso_varying_string

Used by: program tests_excursion_sets

module galacticus_hdf5

subroutine galacticus_version_output

module merger_tree_read

module merger_tree_output_structure

subroutine merger_tree_data_structure.export_galacticus

subroutine store_unit_attributes.galacticus

module galacticus_nodes

subroutine node_component_dark_matter_.profile_scale_shape_tree_output

subroutine stellar_tracks_initialize_.file

subroutine stellar_population_spectra_.file_read

subroutine excursion_sets_first_.crossing_farahi_write_file

module input_parameters

**Type: hdf5object**

Description: A structure that holds properties of HDF5 objects.

Code lines: 250

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

**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_error
galacticus_display
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
Modules used: galacticus_error

function: io_hdf5_dataset_size
Description: Return the size of the dimth dimension of dataset datasetObject.
Code lines: 63
Contained by: module io_hdf5
18. Source Code Documentation

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: 23
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
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
18.1. Program units

**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:** 112

**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: 110
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
Modules used: galacticus_error memory_management

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
18.1. Program units

Modules used: galacticus_error

subroutine: io_hdf5_read_attribute_varstring_1d_array_allocatable
Description: Open and read an 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 an 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 an 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 an 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 an 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 an 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

subroutine: 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

subroutine: 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
18. Source Code Documentation

Modules used: galacticus_error

subroutine: 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

subroutine: 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

subroutine: 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

subroutine: io_hdf5_read_dataset_double_2d_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_3d_array_allocatable
Description: Open and read a double 3-D array dataset in thisObject.
Code lines: 281
Contained by: module io_hdf5
Modules used: galacticus_error memory_management

subroutine: io_hdf5_read_dataset_double_3d_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_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
18.1. Program units

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
Contained by: module io_hdf5
Modules used: galacticus_error

subroutine: io_hdf5_read_dataset_varstring_1d_array_allocatable
Description: Open and read a varying string 1-D array dataset in thisObject.
Code lines: 84
Contained by: module io_hdf5
Modules used: galacticus_error

subroutine: io_hdf5_read_dataset_varstring_1d_array_allocatable_do_read
Description: Open and read a varying string 1-D array dataset in thisObject by creating a suitably-sized character variable into which it can be read.
Code lines: 21

1311
18. Source Code Documentation

**Contained by:** module *io_hdf5*

**Modules used:** memory_management

**subroutine:** io_hdf5_read_dataset_varstring_1d_array_static

**Description:** Open and read an varying string 1-D array dataset in *thisObject*.

**Code lines:** 84

**Contained by:** module *io_hdf5*

**Modules used:** galacticus_error

**subroutine:** io_hdf5_read_dataset_varstring_1d_array_static_do_read

**Description:** 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.

**Code lines:** 17

**Contained by:** module *io_hdf5*

**subroutine:** io_hdf5_set_defaults

**Description:** Sets the compression level and chunk size for dataset output.

**Code lines:** 19

**Contained by:** module *io_hdf5*

**Modules used:** galacticus_error

**subroutine:** io_hdf5_uninitialize

**Description:** Uninitialize the HDF5 subsystem.

**Code lines:** 15

**Contained by:** module *io_hdf5*

**Modules used:** galacticus_error

**subroutine:** io_hdf5_write_attribute_character_1d

**Description:** Open and write an character 1-D array attribute in *thisObject*.

**Code lines:** 90

**Contained by:** module *io_hdf5*

**Modules used:** galacticus_error

**subroutine:** io_hdf5_write_attribute_character_scalar

**Description:** Open and write an character scalar attribute in *thisObject*.

**Code lines:** 96

**Contained by:** module *io_hdf5*

**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*
18.1. Program units

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

1313
18. Source Code Documentation

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
18.1. Program units

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_state_cie_file_read
function electron_density_cie_file_interpolate
function cooling_function_cie_file_interpolate
function cooling_function_cie_file_logtemperature_interpolate
subroutine galacticus_time_per_tree_file_initialize
subroutine galacticus_merger_tree_output_filter_lightcone_initialize
subroutine filter_response_load
subroutine merger_tree_build_initialize
function imf_energy_input_rate_noninstantaneous
function imf_metal_yield_rate_noninstantaneous
function imf_recycling_rate_noninstantaneous
subroutine supernovae_population_iii_hegerwoosley_initialize
subroutine critical_overdensity_mass_scaling_wdm_finalize
subroutine transfer_function_file_read
module input_parameters

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.
18. Source Code Documentation

Code lines: 21
Contained by: module io_xml
Modules used: fox_dom

**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

**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

**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

**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

**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
18.1. Program units

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: 361

module: array_utilities

description: Contains routines which implement useful operations on arrays.
code lines: 341
contained by: file utility.arrays.F90
used by: subroutine merger_tree_data_structure_export_irate
        subroutine table_1d_reverse
        subroutine critical_overdensity_initialize
        subroutine radiation_igb_file_initialize
        program test_array_monotonicity
        program test_make_ranges
        program test_nodes

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.
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

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: 4
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
18.1. Program units

Contained by: module `array_utilities`

**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`

**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 `millennium_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_millennium_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: 104

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: 83
Contained by: file utility.files.F90
Modules used: iso_varying_string
Used by: subroutine galacticus_build_output subroutine galacticus_version_output subroutine merger_trees_millennium_process subroutine merger_tree_write subroutine merger_trees_simple_process
subroutine merger_trees_render_dump subroutine merger_tree_data_structure_-read_ascii
subroutine merger_tree_data_structure_-read_particles_ascii function imf_energy_input_rate_-noninstantaneous
function imf_metal_yield_rate_-noninstantaneous subroutine stellar_population_spectra_-conroy_initialize_imf subroutine excursion_sets_first_-crossing_farahi_read_file
function power_spectrum_nonlinear_-cosmicemu program test_zhao2009_flat
program test_zhao2009_dark_energy program test_zhao2009_open
program tests_halo_mass_function_-tinker 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 existance of file FileName (version for character argument).
Code lines: 7
Contained by: module file_utilities

function: file_exists_varstr
Description: Checks for existance of file FileName (version for varying string argument).
Code lines: 7
Contained by: module file_utilities

file: utility.hashes.F90
Description: Contains a module which implements “hashes” (i.e. associative arrays).
Code lines: 379
module: hashes
Description: Implements “hashes” (i.e. associative arrays).
Code lines: 359
Contained by: file utility.hashes.F90
Modules used: iso_varying_string
Used by: module galacticus_meta_evolver_profiler
program test_hashes

subroutine: delete_integer_scalar_ch
Description: Deletes entry key from thisHash.
Code lines: 10
Contained by: module hashes

subroutine: delete_integer_scalar_vs
Description: Deletes entry key from Hash.
Code lines: 21
Contained by: module hashes
Modules used: arrays_search galacticus_error

function: exists_integer_scalar_ch
Description: Returns true if the specified key exists in the specified thisHash, false otherwise.
Code lines: 10
Contained by: module hashes

function: exists_integer_scalar_vs
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: 82
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: 67
Contained by: module hashes
Modules used: arrays_search

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: 19
Contained by: module hashes
Modules used: arrays_search galacticus_error

subroutine: 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
18. Source Code Documentation

- **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:** 306

**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:** 286

- **Contained by:** file `utility.hashes.perfect.F90`
- **Modules used:** `kind_numbers`
- **Used by:** program `test_perfect_hashes`

**Subroutine:** 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`

**Subroutine:** hash_perfect_destroy

**Description:** Destroy a perfect hash.

**Code lines:** 11

- **Contained by:** module `hashes_perfect`
- **Modules used:** `memory_management`

**Function:** hash_perfect_index

**Description:** Return the index corresponding to a hash key.

**Code lines:** 14

- **Contained by:** module `hashes_perfect`
- **Modules used:** `galacticus_error`

**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
18.1. Program units

- **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:** 1178

**module: input_parameters**
- **Description:** Implements reading of parameters from an XML data file.
- **Code lines:** 1158
- **Contained by:** file utility.input_parameters.F90
- **Modules used:**
  - fox_dom
  - galacticus_versioning
  - io_hdf5
  - iso_c_binding
  - hashes_perfect
  - hashes_cryptographic
  - io_xml
  - iso_varying_string

**Used by:**
- program tests_excursion_sets
- program halo_mass_functions
- program power_spectra
- subroutine accretion_halos_initialize
- subroutine accretion_disks_adaf_get_parameters
- subroutine accretion_disks_initialize
- subroutine accretion_disks_eddington_initialize
- subroutine accretion_disks_switched_initialize
- subroutine atomic_cross_section_ionization_photo_initialize
- subroutine atomic_rate_recombination_radiative_initialize
- subroutine black_hole_binary_initial_radii_spheroid_size_initialize
- function black_hole_binary_initial_radius
- function black_hole_binary_recoil_velocity
- subroutine black_hole_binary_separation_growth_rate
- subroutine black_hole_binary_merger
- subroutine chemical_reaction_rates_initialize
- subroutine chemical_state_cie_file_initialize
subroutine chemical_state_initialize
subroutine cooling_function_initialize
subroutine cooling_radius_output_initialize
subroutine cooling_rate_output_initialize
subroutine cooling_rate_modifier_cutoff
subroutine cooling_rate_simple_scaling_initialize
subroutine cooling_rate_simple_initialize
subroutine cooling_time_initialize
subroutine cooling_freefall_time_available_initialize
subroutine cooling_specific_angular_momentum_initialize
subroutine cooling_time_available_initialize
subroutine cosmology_functions_initialize
module cosmological_parameters
subroutine dark_matter_mass_accretion_wechsler2002_initialize
subroutine halo_spin_distribution_bett2007_initialize
subroutine halo_spin_distribution_delta_function_initialize
subroutine dark_matter_profile_initialize
subroutine dark_matter_concentrations_nfw1996_initialize
subroutine dark_matter_concentrations_prada2011_initialize
subroutine dark_matter_shapes_initialize
subroutine galactic_dynamics_bar_instabilities_eln_initialize
subroutine galactic_structure_radii_solve
subroutine galactic_structure_radii_fixed_initialize
subroutine galactic_structure_initial_radiusadiabatic_initialize
subroutine galacticus_time_per_tree_initialize
subroutine cooling_function_cie_file_initialize
subroutine cooling_radius_initialize
subroutine cooling_rate_initialize
subroutine cooling_rate_white_frenkoff
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 cosmology_functions_matter_dark_energy_initialize
subroutine dark_matter_mass_accretion_initialize
subroutine dark_matter_halo_mass_loss_rates_initialize
function halo_spin_distribution_sample
subroutine halo_spin_distribution_lognormal_initialize
subroutine dark_matter_concentrations_initialize
subroutine dark_matter_concentrations_prada2011_initialize
subroutine node_promotion_index_shift
subroutine galacticus_time_per_tree_file_initialize
18.1. Program units

subroutine galacticus_meta_evolver_profile
subroutine galacticus_output_close_file
subroutine galacticus_output_tree_output
subroutine galacticus_output_tree_descendants_initialize
subroutine galacticus_output_tree_output_filter_lightcone_initialize
subroutine galacticus_output_tree_output_filter_stellar_mass_initialize
subroutine galacticus_output_halo_model_initialize
subroutine galacticus_output_tree_mass_profile_initialize
subroutine galacticus_output_tree_rotation_curve_initialize
subroutine galacticus_output_star_formation_histories_initialize
subroutine galacticus_output_tree_velocity_dispersion_initialize
subroutine output_times_initialize
function galacticus_task_start
subroutine halo_mass_function_compute
subroutine conditional_stellar_mass_functions_initialize
subroutine hot_halo_density_cored_isothermal_core_radii_initialize
subroutine hot_halo_density_cored_isothermal_core_radii_vf_initialize
function hot_halo_ram_pressure_stripping_radius
subroutine hot_halo_temperature_initialize
subroutine intergalactic_medium_state_recfast_initialize
subroutine tree_branching_initialize
subroutine modified_press_schechter_branching_initialize
subroutine merger_tree_branching_modifiers_parkinson_initialize
subroutine merger_tree_build_cole2000_initialize
subroutine meta_tree_timing_initialize
subroutine galacticus_output_open_file
subroutine galacticus_output_tree_density_contrast_initialize
subroutine galacticus_output_tree_output_filter_initialize
subroutine galacticus_output_tree_output_filter_luminosity_initialize
subroutine galacticus_output_tree_output_filter_half_light_initialize
subroutine galacticus_output_tree_output_main_branch_initialize
subroutine galacticus_output_most_massive_progenitor_initialize
subroutine galacticus_output_tree_satellite_pericenter_initialize
subroutine star_formation_histories_metallcity_split_initialize
subroutine galacticus_output_tree_virial_initialize
subroutine state_initialize
function galacticus_task_evolve_tree
subroutine conditional_stellar_mass_functions_behroozi2010_initialize
subroutine hot_halo_density_initialize
subroutine hot_halo_density_cored_isothermal_core_radii_gc_initialize
function hot_halo_ram_pressure_force
subroutine hot_halo_ram_pressure_stripping_font2008_initialize
subroutine intergalactic_medium_state_file_initialize
subroutine intergalactic_medium_state_recfast_initialize
subroutine generalized_press_schechter_branching_initialize
subroutine tree_branching_modifiers_initialize
function merger_tree_create
subroutine merger_tree_build_initialize
subroutine merger_tree_construct_fully_specified_initialize
subroutine merger_trees_mass_function_sampling_gaussian_initialize
subroutine merger_trees_mass_function_sampling_stellar_mf_initialize
subroutine merger_tree_smooth_accretion_initialize
function evolve_to_time
subroutine events_node_merger
subroutine merger_tree_timestep_history
subroutine merger_tree_timestep_satellite
subroutine merger_trees_simple_process
subroutine merger_tree_monotonic_mass_growth
subroutine merger_tree_prune_branches
subroutine merger_tree_regrid_time
subroutine abundances_initialize
function pseudo_random_get
subroutine galacticus_nodes_initialize
subroutine node_component_black_hole_simple_initialize
subroutine node_component_dark_matter_profile_scale_initialize
subroutine node_component_disk_exponential_initialize
subroutine node_component_disk_very_simple_initialize
subroutine node_component_hot_halo_standard_initialize
subroutine node_component_merging_statistics_recent_initialize
subroutine node_component_merging_statistics_standard_initialize
subroutine node_component_satellite_standard_initialize
subroutine node_component_spheroid_standard_initialize
subroutine node_component_spin_random_initialize
subroutine radiation_initialize_intergalactic_background
function ram_pressure_stripping_mass_loss_rate_disk
function dynamical_friction_timescale_multiplier
subroutine merger_trees_mass_function_sampling_initialize
subroutine merger_trees_mass_function_sampling_power_law_initialize
subroutine merger_tree_read_initialize
subroutine merger_tree_state_store_initialize
subroutine merger_tree_evolve_to
subroutine tree_node_evolve_initialize
subroutine merger_tree_timestep_record_evolution
subroutine merger_tree_timestep_simple
subroutine merger_tree_mass_accretion_history_output
subroutine merger_tree_structure_output
subroutine merger_tree_prune_hierarchy
subroutine chemical_abundances_initialize
subroutine node_component_black_hole_simple_initialize
subroutine node_component_dark_matter_profile_scale_shape_initialize
subroutine node_component_disk_exponential_initialize
subroutine node_component_merging_statistics_recent_initialize
subroutine node_component_merging_statistics_standard_initialize
subroutine node_component_satellite_standard_initialize
subroutine node_component_spheroid_standard_initialize
subroutine power_spectrum_compute
subroutine radiation_igb_file_initialize
subroutine ram_pressure_stripping_mass_loss_rate_disks_simple_init
subroutine satellite_merging_mass_movementsbaugh2005_initialize
18.1. Program units

subroutine satellite_merging_mass_movement
subroutine satellite_merging_remnant_sizes_cole2000_initialize
subroutine satellite_merging_remnant_size
subroutine satellite_merging_timescales_initialize
subroutine virial_orbital_parameters_fixed_initialize
subroutine star_formation_imf_initialize
subroutine star_formation_imf_initialize_kennicutt
subroutine star_formation_imf_initialize_millerscalo
subroutine star_formation_imf_initialize_scalo
subroutine imf_select_disk_spheroid_initialize
subroutine star_formation_feedback_disks_courbes2012_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_feedback_spheroids_sw_initialize
subroutine star_formation_feedback_spheroids_sw_initialize
subroutine star_formation_feedback_spheroids_sw_initialize
subroutine star_formation_rate_surface_density_disks_br_initialize
subroutine star_formation_rate_surface_density_disks_ks_initialize
subroutine star_formation_timescale_disks_initialize
subroutine star_formation_timescale_disks_fixed_initialize
subroutine star_formation_timescale_spheroids_initialize
subroutine star_formation_timescale_spheroids_fixed_initialize
subroutine star_formation_timescale_spheroids_dynamical_time_initialize

subroutine satellite_merging_mass_movements_simple_initialize
subroutine satellite_merging_remnant_sizes_covington2008_initialize
subroutine satellite_merging_remnant_progenitor_properties
function virial_orbital_parameters

subroutine star_formation_imf_initialize_baugh2005topheavy
subroutine star_formation_imf_initialize_chabrier
subroutine star_formation_imf_initialize
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_halo_scaling_initialize
subroutine star_formation_feedback_disks_power_law_initialize
subroutine star_formation_feedback_spheroids_initialize
subroutine star_formation_feedback_spheroids_halo_scaling_initialize
subroutine star_formation_feedback_spheroids_power_law_initialize
subroutine star_formation_expulsive_feedback_disks_initialize
subroutine star_formation_expulsive_feedback_spheroids_initialize
subroutine star_formation_expulsive_feedback_spheroids_sw_initialize
subroutine star_formation_rate_surface_density_disks_kmt09_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
subroutine star_formation_timescale_spheroids_dynamical_time_initialize
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_properties_noninstantaneous_initialize

subroutine stellar_population_spectra_file_initialize_imf
subroutine stellar_population_spectra_postprocess_recent
subroutine critical_overdensity_mass_scaling_initialize
subroutine excursion_sets_barrier_initialize
subroutine excursion_sets_quadratic_initialize
subroutine excursion_sets_first_crossings_initialize
subroutine dark_matter_halo_bias_initialize
subroutine linear_growth_initialize

function power_spectrum_nonlinear_cosmicemu
subroutine primordial_power_spectrum_power_law_initialize
subroutine power_spectrum_window_functions_sharp_kspace_initialize
subroutine sphericalCollapseDarkEnergyDeltaVirial_initialize
subroutine transfer_function_cmbfast_make
subroutine transfer_function_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_luminosities_initialize
subroutine stellar_population_spectrum_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 initialize_cosmological_mass_variance
subroutine power_spectrum_nonlinear_cosmicemu_initialize
subroutine power_spectrum_initialize

subroutine primordial_power_spectrum_power_law_initialize
subroutine power_spectrum_window_functions_initialize
subroutine power_spectrum_window_functions_th_kss_hybrid_initialize
subroutine transfer_function_bbks_initialize
subroutine transfer_function_eisenstein_hu_initialize
subroutine transfer_function_file_initialize
18.1. Program units

subroutine virial_density_contrast_initialize
program test_nfw96_concentration_dark_energy
program test_zhao2009_flat
dark_energy
program test_zhao2009_open
program tests_bug745815
program tests_comoving_distance_dark_energy
program tests_comoving_distance_fixed_open
program test_zhao2009_dark_energy
program test_abundances
program tests_comoving_distance_eds
program tests_comoving_distance_open
program test_zhao2009_flat
program test_zhao2009_dark_energy
program test_zhao2009_open
program test_zhao2009_dark_energy
program test_zhao2009_open
program test_abundances
program tests_comoving_distance_eds
program tests_comoving_distance_open

subroutine virial_density_fixed_initialize
program test_prada2011_concentration

program tests_cosmic_age_eds
program tests_cosmic_age_dark_energy_closed
program tests_cosmic_age_dark_energy_omega_minus_one_third
program tests_halo_mass_function_tinker
program tests_linear_growth_eds
program tests_linear_growth_dark_energy
program test_nodes
program tests_sigma
program tests_power_spectrum
program tests_sphericalCollapseDarkEnergyEds
program tests_sphericalCollapseDarkEnergyLambda
program tests_sphericalCollapseFlat

program tests_sphericalCollapseDarkEnergyOmegaZeroPointSix
program tests_sphericalCollapseDarkEnergyOmegaZeroPointEight
program tests_sphericalCollapseDarkEnergyOmegaHalf

subroutine close_parameters_group
Code lines: 7
Contained by: module input_parameters

interface: get_input_parameter
Code lines: 13
Contained by: module input_parameters

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: 58
Contained by: module input_parameters

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: 66
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: 58
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: 58
Contained by: module input_parameters

subroutine: get_input_parameter_double_c

Description: C-bound wrapper function for getting double precision parameter values.

Code lines: 13
Contained by: module input_parameters
Modules used: string_handling

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.)

**Code lines:** 58

**Contained by:** module `input_parameters`

**subroutine:** `get_input_parameter_integer_array`
**Source Code Documentation**

*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.)

*Code lines:* 58
*Contained by:* module `input_parameters`

**subroutine:** `get_input_parameter_integer_c`

*Description:* C-bound wrapper function for getting **integer** parameter values.
*Code lines:* 13
*Contained by:* module `input_parameters`  
*Modules used:* `string_handling`

**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.)

*Code lines:* 59
*Contained by:* module `input_parameters`  
*Modules used:* `kind_numbers`

**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.)

*Code lines:* 59
*Contained by:* module `input_parameters`  
*Modules used:* `kind_numbers`

**subroutine:** `get_input_parameter_logical`

*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:* 65
*Contained by:* module `input_parameters`  

**subroutine:** `get_input_parameter.logical_array`

1334
**18.1. Program units**

*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:* 64

*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:** 58

**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:** 66

**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>
  <parameter>
    <name>parameter1Name</name>
    <value>parameter1Value</value>
  </parameter>
  <parameter>
    <name>parameter1Name</name>
    <value>parameter1Value</value>
  </parameter>
  .
  .
  .
</parameters>
```

**Code lines:** 116

**Contained by:** module input_parameters
18.1. Program units

Modules used:
- file_utilities
- galacticus_display
- galacticus_input_paths
- string_handling

**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*: 58
- *Contained by*: module input_parameters

**function**: transfer_function cmbfast_label
- *Code lines*: 30
- *Contained by*: module input_parameters

**subroutine**: write_parameter
- *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 cooling_radius_hot_halo_output
- *Module*: cooling_radiiIsothermal
- module cooling_radii_isothermal
- module cooling_radii_simple
- module cooling_specific_angular_momenta_constant_rotation
- module dark_matter_profiles_nfw
- subroutine galacticus_output_tree_density_contrast
- subroutine galacticus_output_tree_lightcone
- subroutine galacticus_extra_output_halo_fourier_profile
- subroutine galacticus_output_tree_links
- subroutine galacticus_output_tree_mass_profile
- subroutine galacticus_output_tree_rotation_curve
- subroutine galacticus_output_tree_descendants
- subroutine galacticus_output_tree_half_light
- subroutine galacticus_output_tree_descendants
- subroutine galacticus_output_tree_half_light
- subroutine galacticus_output_tree_massive_progenitor
- subroutine galacticus_output_tree_satellite_pericenter
- subroutine galacticus_output_tree_main_branch
- subroutine galacticus_output_tree_massive_progenitor
- subroutine galacticus_output_tree_satellite_pericenter

1337
module galacticus_output_star_formation_histories
subroutine star_formation_history_output_null
subroutine galacticus_output_tree_velocity_dispersion
subroutine merger_tree_build_do_cole2000
subroutine merger_tree_construct_fully_specified
function node_definition_index
module merger_tree_read
module merger_trees_state_store
module merger_trees_evolve
module merger_trees_evolve_node
module numerical_constants_math
function sort_index_do_integer8
subroutine kepler_orbits_output
module merger_trees_dump
subroutine node_component_black_hole_simple_output
subroutine node_component_black_hole_standard_output
subroutine node_component_black_hole_standard_output_properties
subroutine node_component_disk_exponential_star_formation_history_output
module node_component_disk_exponential_data
subroutine node_component_inter_output_standard_reset
subroutine node_component_spheroid_standard_star_formation_history_output
module star_formation_rate_surface_density_disks_br
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_timescale_disks_halo_scaling
function excursion_sets_barrier_effective
subroutine excursion_sets_first_crossing_rate_tabulate_farahi
program tests_io_hdf5
program tests_bug745815
program test_search
program tests_tree_branch_destroy
subroutine io_hdf5_read_attribute_integer8_1d_array_static
subroutine star_formation_history_output_metallicity_split
subroutine galacticus_output_tree_velocity_dispersion
subroutine merger_tree_build_do_cole2000
function node_lookup
subroutine merger_tree_smooth_accretion_do
module merger_trees_evolve
subroutine evolve_to_time_report
module merger_trees_render
subroutine search_array_integer8
subroutine sort_index_do_integer8_c
module merger_tree_data_structure
subroutine node_component_black_hole_simple_output
subroutine node_component_black_hole_standard_output_properties
module node_component_disk_exponential_data
subroutine node_component_merging_statistics_recent_output
module star_formation_rate_surface_density_disks_br
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_timescale_disks_halo_scaling
function excursion_sets_first_crossing_probability_farahi
subroutine make_table
program test_array_monotonicity
program test_perfect_hashes
program test_string_utilities
subroutine io_hdf5_read_attribute_integer8_1d_array_allocatable
subroutine io_hdf5_read_attribute_integer8_scalar
1338
subroutine io_hdf5_read_dataset_integer8_1d_array_allocatable
subroutine io_hdf5_read_dataset_integer8_1d_static
subroutine io_hdf5_write_attribute_integer8_1d
subroutine io_hdf5_write_dataset_integer8_1d
function array_index_integer8
module hashes_perfect
subroutine get_input_parameter_integer_long
module memory_management
subroutine assert_integer8_1d_array

file: utility.memory_management.F90

Description: Contains a module for storing and reporting memory usage by the code.
Code lines: 1237

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 Make_Memory_Usage_Routines.pl. Contains memory management subroutines. This file was created automatically by Make_Memory_Usage_Routines.pl.

Code lines: 1217

Contained by: file utility.memory_management.F90

Modules used:
- iso_c_binding
- kind_numbers
- program tests_excursion_sets
- program halo_mass_functions
- program millennium_merge_tree_file-maker
- program power_spectra
- subroutine atomic_data_initialize
- subroutine chemical_state_cie_file_read
- subroutine cooling_function_initialize
- subroutine make_expansion_factor_table
- subroutine make_distance_table
- subroutine dark_profile_einsto_freefall_tabulate
- subroutine fourier_profile_table_make
- subroutine galacticus_meta_evolver_profile
- subroutine allocate_buffers
- subroutine galacticus_output_tree_density_contrast_initialize
- subroutine galacticus_merge_tree_output_filter_lightcone_initialize
- subroutine galacticus_extra_output_halo_fourier_profile

Used by:
- program optimal_sampling_smf
- program simple_merge_tree_file_maker
- subroutine chemical_reaction_rates_initialize
- subroutine cooling_function_cie_file_read
- subroutine cosmology_matter_dark_energy_state_retrieve
- subroutine cosmology_matter_lambda_state_retrieve
- subroutine make_expansion_factor_table
- subroutine energy_table_make
- subroutine radius_from_specific_angular_momentum_table_make
- subroutine meta_tree_timing_post_evolve
- subroutine make_output_group
- subroutine galacticus_merge_tree_output_filter_initialize
- subroutine galacticus_merge_tree_output_filter_luminosity_initialize
18.1. Program units

subroutine galacticus_linear_power_spectrum_output
subroutine star_formation_histories_metallicity_split_initialize
subroutine output_times_initialize
subroutine halo_mass_function_compute
function merger_tree_create
subroutine merger_tree_constructfully_specified
subroutine create_node_array
subroutine destroy_node_indices
subroutine merger_tree_read_initialize
subroutine tree_node_evaluate
subroutine merger_tree_timestep_record_evolution
subroutine merger_tree_structure_output
subroutine merger_trees_read_dump
function interpolate_2d_irregular_array

subroutine abundances_destroy
subroutine chemical_abundances_allocate_values
subroutine chemicals_abundances_destroy
subroutine history_clone
subroutine history_destroy
subroutine history_timesteps
subroutine merger_tree_data_constructparticle_indices
subroutine merger_tree_data_structureexport_galacticus
subroutine merger_tree_data_structureread_ascii
subroutine merger_tree_data_structurereset
subroutine merger_tree_data_structureset_property_column
subroutine merger_tree_data_structureset_property_double
subroutine merger_tree_data_structureset_property_integer8
module galacticus_nodes

subroutine basicnon-evolvingtimelastisolatedset
subroutine basicstandardaccretionrateset
subroutine basicstandardtimelastisolatedset
subroutine galacticus_output_tree_mass_profile_initialize
subroutine star_formation_history_scales_metallicity_split
function galacticus_task_evaluate_tree
subroutine filter_response_load
subroutine merger_tree_build_initialize
subroutine build_child_and_sibling-links
subroutine create_node_indices
subroutine merger_tree_read_do
subroutine read_node_data
subroutine merger_tree_timestep_history
subroutine merger_tree_mass_accretion_history_output
subroutine merger_tree_regrid_time
subroutine merger_tree_write
subroutine abundances_allocateelemental_values
subroutine abundances_initialize
subroutine chemical_abundances_initialize
subroutine chemicals_read_raw
subroutine history_create
subroutine history_read_raw
subroutine history_trim
subroutine merger_tree_data_structureadd_metadata
subroutine merger_tree_data_structureexport_IRATE
subroutine merger_tree_data_structureread_particles_ascii
subroutine merger_tree_data_structurereset
subroutine merger_tree_data_structureset_particle_property_column
subroutine merger_tree_data_structureset_property_column
subroutine merger_tree_data_structureset_property_double
subroutine merger_tree_data_structureset_tree_indices
subroutine basicnon-evolvingmassset
subroutine basicstandardmassset
subroutine basicnon-evolvingtimeset
subroutine basicstandardmassset
subroutine basicstandardtimeset
subroutine **blackholesimplemassset** subroutine **blackholestandardmassset** subroutine **blackholestandardradialpositionset** subroutine **blackholestandardtripleinteractiontimeset** subroutine **darkmatterprofilescalescaleset** subroutine **darkmatterprofilescalescaleshapeset** subroutine **deserializefromarrayscales** subroutine **deserializefromarrayvalues** subroutine **diskexponentialabundancesgasset** subroutine **diskexponentialabundancesstellarset** subroutine **diskexponentialangularmomentumset** subroutine **diskexponentialalluminositiesstellarset** subroutine **diskexponentialmassstellarset** subroutine **diskexponentialmassstellarset** subroutine **diskexponentialstarformationhistoryset** subroutine **diskexponentialstellarpropertieshistoryset** subroutine **diskverysimplemassgasset** subroutine **diskverysimplemassstellarset** subroutine **diskverysimplemassstellarset** subroutine **diskverysimplemassstellarset** subroutine **diskverysimplemassgasset** subroutine **diskverysimplemassstellarset** subroutine **galacticus_nodes_initialize** subroutine **hothalostandardabundancesset** subroutine **hothalostandardangularmomentumset** subroutine **hothalostandardchemicalsset** subroutine **hothalostandardaccelerationset** subroutine **hothalostandardouterradiusset** subroutine **hothalostandardoutflowedabundancesset** subroutine **hothalostandardoutflowedangularmomentumset** subroutine **hothalostandardoutflowedmassset** subroutine **hothalostandardstrippedabundancesset** subroutine **hothalostandardstrippedmassset** subroutine **hothalostandardunaccretedmassset** subroutine **hothaloverysimplemassset** subroutine **hothalrustandardmassset** subroutine **hothalustandardmasssstellarset** subroutine **interoutputstandarddiskstarformationrateset** subroutine **interoutputstandardspheroidstarformationrateset** subroutine **mergingstatisticsrecentrecentmajormergercountset** subroutine **mergingstatisticsrecentrecentmajormergercountset** subroutine **mergingstatisticsstandardnodehierarchylevelset** subroutine **mergingstatisticsstandardnodehierarchylevelset** subroutine **node_component_-basicnonevolving_builder** subroutine **node_component_-basicnonevolving_destroy** subroutine **node_component_-basicnonevolving_initializor** subroutine **node_component_-basicnonevolving_read_raw**
subroutine node_component_basicnull_builder
subroutine node_component_basicnull_destroy
subroutine node_component_basicnull_initializor
subroutine node_component_basicstandard_builder
subroutine node_component_basicstandard_destroy
subroutine node_component_basicstandard_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_darkmatterprofilescaleshape_builder
subroutine node_component_darkmatterprofilescaleshape_destroy
subroutine node_component_darkmatterprofilescaleshape_initializor
subroutine node_component_disknull_builder
subroutine node_component_disknull_destroy
subroutine node_component_disknull_initializor
subroutine node_component_diskverysimple_builder
subroutine node_component_diskverysimple_destroy
subroutine node_component_diskverysimple_initializor
1343
18.1. Program units

subroutine node_component_-mergingstatisticsstandard_builder
subroutine node_component_-mergingstatisticsstandard_initializor
subroutine node_component_-positionnull_builder
subroutine node_component_-positionnull_initializor
subroutine node_component_-positionpreset_builder
subroutine node_component_-positionpreset_initializor
subroutine node_component_-satellitenull_builder
subroutine node_component_-satellitenull_initializor
subroutine node_component_-satellitepreset_builder
subroutine node_component_-satellitepreset_initializor
subroutine node_component_-satellitestandard_builder
subroutine node_component_-satellitestandard_initializor
subroutine node_component_-satelliteverysimple_builder
subroutine node_component_-satelliteverysimple_initializor
subroutine node_component_-spheroidnull_builder
subroutine node_component_-spheroidnull_initializor
subroutine node_component_-spheroidstandard_builder
subroutine node_component_-spheroidstandard_initializor
subroutine node_component_spinnull_-builder
subroutine node_component_spinnull_-initializor
subroutine node_component_spinpreset_-builder
subroutine node_component_spinpreset_-initializor
subroutine node_component_-mergingstatisticsstandard_destroy
subroutine node_component_-mergingstatisticsstandard_read_raw
subroutine node_component_-positionnull_destroy
subroutine node_component_-positionnull_read_raw
subroutine node_component_-positionpreset_destroy
subroutine node_component_-positionpreset_read_raw
subroutine node_component_-satellitenull_destroy
subroutine node_component_-satellitenull_read_raw
subroutine node_component_-satellitepreset_destroy
subroutine node_component_-satellitepreset_read_raw
subroutine node_component_-satellitestandard_destroy
subroutine node_component_-satellitestandard_read_raw
subroutine node_component_-satelliteverysimple_destroy
subroutine node_component_-satelliteverysimple_read_raw
subroutine node_component_-spheroidnull_destroy
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_spinpreset_-destroy
subroutine node_component_spinpreset_-read_raw
subroutine node_component_spinrandom_-builder
subroutine node_component_spinrandom_-initializor
subroutine positionpresetpositionhistoryset
subroutine positionpresetpositionset
subroutine positionpresetvelocityset
subroutine satellitepresetboundmasshistoryset
subroutine satellitepresetvirialorbitset
subroutine satellitepresettimeofmergingset
subroutine satellitestandardboundmassset
subroutine satellitestandardmergetimeset
subroutine satellitestandardvirialorbitsetvalue
subroutine serializetoarrayrates
subroutine serializetoarrayvalues
subroutine spheroidstandardabundancesgasset
subroutine spheroidstandardabundancesstellarset
subroutine spheroidstandardangularmomentumset
subroutine spheroidstandardisinitializedset
subroutine spheroidstandardluminositiesstellarset
subroutine spheroidstandardmassgasset
subroutine spheroidstandardmassstellarset
subroutine spheroidstandardradiusset
subroutine spheroidstandardstellarpropertieshistoryset
subroutine spinpresentspinscale
subroutine spinpresentspinset
subroutine node_component_disk_-exponential_initialize
subroutine node_component_spheroid_-standard_initialize
subroutine table_1d_destroy
subroutine table_linear_1d_create
subroutine table_linear_cspline_1d_-create
subroutine table_linear_cspline_1d_-destroy
subroutine radiation_set.cmb
subroutine radiation_igb_file_-initialize
subroutine radiation_igb_file_set
function imf_energy_input_rate_-noninstantaneous
function imf_metal_yield_rate_-noninstantaneous
function imf_recycling_rate_-noninstantaneous
subroutine star_formation_imf_initialize
subroutine stellarAstrophysicsFileInitialize
subroutine stellar_tracks_initialize_file
subroutine stellar_population_properties_initialize
subroutine stellar_population_spectra_conroy_initialize_imf
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
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_hdf5
program test_prada2011_concentration
program test_zhao2009_dark_energy
program test_abundances
program tests_comoving_distance_eds
program tests_comoving_distance_open
program tests_cosmic_age_-cosmological_constant
program tests_cosmic_age_-dark_energy_-cosmological_constant
program tests_cosmic_age_open
program test_hashes
program test_perfect_hashes
program tests_linear_growth_eds
subroutine star_formation_imf_initialize_piecewisepowerlaw
subroutine supernovae_type_ia_nagashima_initialize
function stellar_population_luminosity
subroutine stellar_population_properties_scales_noninstantaneous
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_state_retrieve
subroutine transfer_function_null_make
program tests_comoving_distance_dark_energy
program tests_halo_mass_function_tinker
program test_hashes_cryptographic
program tests_kepler_orbits
program tests_linear_growth_-cosmological_constant
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: 14
18.1. Program units

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

subroutine: alloc_array_double_precision_1d
Description: Allocate a 1D double_precision array.
Code lines: 40
Contained by: module memory_management
deep
Modules used: dmemory iso_varying_string
galacticus_display
string_handling

subroutine: alloc_array_double_precision_2d
Description: Allocate a 2D double_precision array.
Code lines: 40
Contained by: module memory_management
deep
Modules used: dmemory iso_varying_string
galacticus_display
string_handling

subroutine: alloc_array_double_precision_3d
Description: Allocate a 3D double_precision array.
Code lines: 40
Contained by: module memory_management
deep
Modules used: dmemory iso_varying_string
galacticus_display
string_handling

subroutine: alloc_array_double_precision_4d
Description: Allocate a 4D double_precision array.
Code lines: 40
Contained by: module memory_management
deep
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
deep
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
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_kind_int8_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_kind_int8_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_logical_1d
Description: Allocate a 1D logical array.
Code lines: 40
Contained by: module memory_management
Modules used: dmemory iso_varying_string
galacticus_display string_handling

subroutine: alloc_array_real_kind_quad_1d
Description: Allocate a 1D real array.
Code lines: 40
Contained by: module memory_management
Modules used: dmemory iso_varying_string
galacticus_display 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_input_paths iso_varying_string
galacticus_display string_handling

interface: dealloc_array
18.1. Program units

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: 32
Contained by: module memory_management
Modules used: dmemory
                  galacticus_display
                  galacticus_error
                  iso_varying_string
                  string_handling

subroutine dealloc_array_double_precision_1d
Description: Deallocate a 1D double_precision array.
Code lines: 32
Contained by: module memory_management
Modules used: dmemory
                  galacticus_display
                  galacticus_error
                  iso_varying_string
                  string_handling

subroutine dealloc_array_double_precision_2d
Description: Deallocate a 2D double_precision array.
Code lines: 32
Contained by: module memory_management
Modules used: dmemory
                  galacticus_display
                  galacticus_error
                  iso_varying_string
                  string_handling

subroutine dealloc_array_double_precision_3d
Description: Deallocate a 3D double_precision array.
Code lines: 32
Contained by: module memory_management
Modules used: dmemory
                  galacticus_display
                  galacticus_error
                  iso_varying_string
                  string_handling

subroutine dealloc_array_double_precision_4d
Description: Deallocate a 4D double_precision array.
Code lines: 32
Contained by: module memory_management
Modules used: dmemory
                  galacticus_display
                  galacticus_error
                  iso_varying_string
                  string_handling

subroutine dealloc_array_double_precision_5d
Description: Deallocate a 5D double_precision array.
Code lines: 32
Contained by: module memory_management
subroutine: dealloc_array_integer_1d
Description:Deallocate a 1D integer array.
Code lines:32
Contained by:module memory_management
Modules used:dmemory
galacticus_error
iso_varying_string
string_handling
subroutine: dealloc_array_integer_2d
Description:Deallocate a 2D integer array.
Code lines:32
Contained by:module memory_management
Modules used:dmemory
galacticus_error
iso_varying_string
string_handling
subroutine: dealloc_array_integer_kind_int8_1d
Description:Deallocate a 1D integer array.
Code lines:32
Contained by:module memory_management
Modules used:dmemory
galacticus_error
iso_varying_string
string_handling
subroutine: dealloc_array_integer_kind_int8_2d
Description:Deallocate a 2D integer array.
Code lines:32
Contained by:module memory_management
Modules used:dmemory
galacticus_error
iso_varying_string
string_handling
subroutine: dealloc_array_logical_1d
Description:Deallocate a 1D logical array.
Code lines:32
Contained by:module memory_management
Modules used:dmemory
galacticus_error
iso_varying_string
string_handling
subroutine: dealloc_array_real_kind_quad_1d
Description:Deallocate a 1D real array.
Code lines:32
Contained by:module memory_management
### 18.1. Program units

**Modules used:**  
dmemory  
galacticus_error  
galacticus_display  
iso_varying_string  
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  
galacticus_display  
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  
galacticus_display  
iso_varying_string

**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.string_handling.F90

- **Description:** Contains a module which implements various useful functionality for manipulating character strings.  
- **Code lines:** 368

**module:** string_handling
18. Source Code Documentation

**Description:** Implements various useful functionality for manipulating character strings.

**Code lines:** 348

**Contained by:** file `utility.string_handling.F90`

**Modules used:** `iso_varying_string`

**Used by:**
- function `abundance_pattern_lookup`
- subroutine `atomic_data_initialize`
- subroutine `cooling_function_not_matched`
- function `node_branch_jump`
- function `node_subhalo_promotion`
- subroutine `solve_for_radius`
- subroutine `galacticus_merge_tree_output_make_group`
- subroutine `galacticus_extra_output_halo_fourier_profile`
- subroutine `star_formation_history_output_metallicity_split`
- function `galacticus_version`
- subroutine `galacticus_state_store`
- subroutine `merger_tree_build_do`
- subroutine `build_descendent_pointers`
- subroutine `build_parent_pointers`
- subroutine `create_node_indices`
- subroutine `merger_tree_read_do`
- subroutine `merger_tree_state_restore`
- subroutine `merger_tree_evolve_to`
- subroutine ` producing_error_handler`
- subroutine `merger_tree_record_evolution_output`
- subroutine `satellite_merger_process`
- subroutine `merger_tree_structure_output`
- subroutine `history_extend`
- subroutine `merger_tree_data_structure_export_galacticus`
- subroutine `merger_tree_data_structure_read_ascii`
- subroutine `blackholecreate_linked`
- subroutine `diskcreate_linked`
- subroutine `hotalocreate_linked`
- function `atom_lookup`
- subroutine `chemical_state_atomic_cie_cldy_create`
- subroutine `cooling_function_atomic_cie_cldy_create`
- subroutine `events_node_merge_do_slh`
- function `galacticus_structure_radius_enclosing_mass`
- subroutine `galacticus_build_output`
- subroutine `make_output_group`
- subroutine `galacticus_output_tree_rotation_curve_initialize`
- subroutine `galacticus_output_tree_velocity_dispersion_initialize`
- subroutine `galacticus_state_retrieve`
- function `galacticus_task_evolve_tree`
- subroutine `assign_mergers`
- subroutine `build_isolated_parent_pointers`
- subroutine `build_subhalo_mass_histories`
- subroutine `enforce_subhalo_status`
- subroutine `scan_for_mergers`
- function `evolve_to_time`
- subroutine `events_node_merge`
- subroutine `tree_node_promote`
- 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 `basiccreate_linked`
- subroutine `darkmatterprofilecreate_linked`
- subroutine `formationtimecreate_linked`
- subroutine `indicescreate_linked`
18.1. Program units

subroutine interoutputcreatelinked
subroutine node_component_basicnonevolving_dump
subroutine node_component_basicstandard_dump
subroutine node_component_blackholesimple_dump
subroutine node_component_darkmatterprofilenull_dump
subroutine node_component_darkmatterprofilescale_dump
subroutine node_component_darkmatterprofilescaleshape_dump
subroutine node_component_disknull_dump
subroutine node_component_diskverysimple_dump
subroutine node_component_formationtimecole2000_dump
subroutine node_component_formationtimenull_dump
subroutine node_component_hothalonull_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_mergingstatisticsnull_dump
subroutine node_component_mergingstatisticsrecent_dump
subroutine node_component_mergingstatisticsstandard_dump
subroutine node_component_positionnull_dump
subroutine node_component_positionpreset_dump
subroutine node_component_satellitenull_dump
subroutine node_component_satellitepreset_dump
subroutine node_component_spheroidnull_dump
subroutine node_component_spheroidstandard_dump
subroutine node_component_spinnull_dump
subroutine node_component_s家庭null_dump
dump

subroutine mergingstatisticscreatelinked
subroutine node_component_basicnull_dump
subroutine node_component_blackholenull_dump
subroutine node_component_blackholestandard_dump
subroutine node_component_darkmatterprofilescaladump
subroutine node_component_darkmatterprofilescaleshape_dump
subroutine node_component_diskverysimple_dump
subroutine node_component_formationtimenull_dump
subroutine node_component_hothalostandard_dump
subroutine node_component_indicesnull_dump
subroutine node_component_interoutputnull_dump
subroutine node_component_mergingstatisticsnull_dump
subroutine node_component_mergingstatisticsstandard_dump
subroutine node_component_positionnull_dump
subroutine node_component_satellitepreset_dump
subroutine node_component_spheroidstandard_dump
subroutine node_component_spinnull_dump
subroutine node_component_spheroidstandard_dump

subroutine positioncreatelinked
subroutine spheroidecreatelinked
subroutine tree_node_remove_from_host
subroutine node_component_black_hole_standard_output_properties
subroutine node_component_disk_exponential_post_evolve
subroutine node_component_disk_very_simple_post_evolve
subroutine satellitecreatelinked
subroutine spincreatelinked
subroutine tree_node_remove_from_mergee
subroutine node_component_disk_exponential_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_cmbfast_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 get_input_parameter_varstring_array
subroutine input_parameters_file_open

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_2d
subroutine alloc_array_integer_kind_int8_1d
subroutine alloc_array_integer_kind_int8_2d
subroutine alloc_array_real_kind_quad_1d
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_2d
subroutine dealloc_array_integer_kind_int8_1d
subroutine dealloc_array_integer_kind_int8_2d
subroutine dealloc_array_logical_1d

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 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
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_logical_1d
18.1. Program units

subroutine dealloc_array_real_kind_quad_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.
Code lines: 13
Contained by: module string_handling
Modules used: iso_c_binding

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_subscript**

- **Description:** Converts an input string to Unicode subscripts.
- **Code lines:** 16
- **Contained by:** module *string_handling*

**function: string_superscript**

- **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_ode_solver*
  - program *

1358
### 18.1. Program units

<table>
<thead>
<tr>
<th>Program</th>
<th>Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>program test_prada2011_concentration</td>
<td>program test_zhao2009_flat</td>
</tr>
<tr>
<td>program test_zhao2009_dark_energy</td>
<td>program test_zhao2009_open</td>
</tr>
<tr>
<td>program test_abundances</td>
<td>program test_array_monotonicity</td>
</tr>
<tr>
<td>program test_black_hole_fundamentals</td>
<td>program tests_bug745815</td>
</tr>
<tr>
<td>program tests_comoving_distance_eds</td>
<td>program tests_comoving_distance_dark_energy</td>
</tr>
<tr>
<td>program tests_comoving_distance_open</td>
<td>program test_comparison</td>
</tr>
<tr>
<td>program tests_cosmic_age_edds</td>
<td>program tests_cosmic_age_cosmological_constant</td>
</tr>
<tr>
<td>program tests_cosmic_age_dark_energy_closed</td>
<td>program tests_cosmic_age_dark_energy_cosmological_constant</td>
</tr>
<tr>
<td>program tests_cosmic_age_dark_energy_omega_minus_one_third</td>
<td>program tests_cosmic_age_open</td>
</tr>
<tr>
<td>program test_coordinate_systems</td>
<td>program tests_halo_mass_function_tinker</td>
</tr>
<tr>
<td>program test_hashes</td>
<td>program test_hashes_cryptographic</td>
</tr>
<tr>
<td>program test_perfect_hashes</td>
<td>program test_integration</td>
</tr>
<tr>
<td>program test_interpolation_2d</td>
<td>program test_interpolation</td>
</tr>
<tr>
<td>program tests_kepler_orbits</td>
<td>program tests_linear_growth_eds</td>
</tr>
<tr>
<td>program tests_linear_growth_omega_minus_one_third</td>
<td>program tests_linear_growth_dark_energy</td>
</tr>
<tr>
<td>program tests_linear_growth_open</td>
<td>program test_make_ranges</td>
</tr>
<tr>
<td>program test_mass_distributions</td>
<td>program test_math_special_functions</td>
</tr>
<tr>
<td>program test_meshes</td>
<td>program test_nodes</td>
</tr>
<tr>
<td>subroutine test_node_task</td>
<td>program tests_power_spectrum</td>
</tr>
<tr>
<td>program test_root_finding</td>
<td>program test_search</td>
</tr>
<tr>
<td>program tests_sigma</td>
<td>program test_sort</td>
</tr>
<tr>
<td>program tests_sphericalCollapse_dark_energy_eds</td>
<td>program tests_sphericalCollapse_dark_energy_omega_zero_point_six</td>
</tr>
<tr>
<td>program tests_sphericalCollapse_dark_energy_with_two_thirds</td>
<td>program tests_spherical Collapse dark energy lambda</td>
</tr>
<tr>
<td>program tests_sphericalCollapse_dark_energy_with_two_thirds</td>
<td>program tests_sphericalCollapse_dark_energy_with_two_thirds</td>
</tr>
<tr>
<td>program tests_sphericalCollapse_dark_energy_with_two_thirds</td>
<td>program tests_sphericalCollapse dark energy lambda</td>
</tr>
<tr>
<td>program tests_sphericalCollapse_dark_energy_with_two_thirds</td>
<td>program tests_sphericalCollapse flat</td>
</tr>
<tr>
<td>program tests_sphericalCollapse_open</td>
<td>program test_string_utilities</td>
</tr>
<tr>
<td>program test_tables</td>
<td>program tests_tree_branch_destroy</td>
</tr>
<tr>
<td>program test_vectors</td>
<td>program test_vectors</td>
</tr>
</tbody>
</table>

**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
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
18.1. Program units

**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 string_handling memory_management
Part V.

Contributions and Acknowledgements
19. Contributions

Contributions to the GALACTICUS project have been made by the following people:

Andrew Benson
- 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
- objects.nodes.components.satellite.preset.F90
- 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

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

Jianling Gan
- objects.nodes.components.satellite.preset.F90

Luiz Felippe S. Rodrigues
- intergalatic_medium.state.RecFast.F90
- intergalatic_medium.state.file.F90

Martin White
- satellites.merging.dynamical_friction.timescale.Wetzel-White.F90
19. Contributions

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
20. 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 users¹ 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.

¹In particular, Jianling Gan, Markus Haider, Ting-Wen Lan, Luiz Felippe Rodrigues, Sergio Sanes, Martin White and Liyan Xu.
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 useable 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" {
```
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 $8h^{-1}\text{Mpc}$ 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): "CMBFast"
    }
  }
}
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 "OfTypeTimestepCriterion" {
        DATATYPE H5T_IEEE_F64LE
        DATASPACE SCALAR
        DATA {
            (0): 0.02
        }
    }
}
A. Merger Tree File Format

```plaintext
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 attributes 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;

**SofteningBndryMaxPhys** 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

- \( \text{lengthUnitsInSI} = 3.08568 \times 10^{22} \)
- \( \text{lengthHubbleExponent} = -1 \)
- \( \text{lengthScaleFactorExponent} = 1 \)

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 "treesAreSelfContained" {
        DATATYPE  H5T_STD_I32LE
        DATASPACE  SCALAR
        DATA {
            (0): 1
        }
    }
    ATTRIBUTE "treesHaveSubhalos" {
        DATATYPE  H5T_STD_I32LE
        DATASPACE  SCALAR
        DATA {
            (0): 1
        }
    }
}
A.6. Halo Trees Group

```
}]

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" {
    COMMENT "The mass of each node."
    DATATYPE H5T_IEEE_F64LE
    DATASPACE SIMPLE { ( <nodeCount> ) / ( <nodeCount> ) }
}

DATASET "particleCount" {
    COMMENT "The number of entries within the particles group for this node."
    DATATYPE H5T_STD_I64LE
```
A. Merger Tree File Format

DATASPACE SIMPLE { ( <nodeCount> ) / ( <nodeCount> ) }

DATASET "particleStart" {
COMMENT "The index within the particles group at which the particle data for this node is stored."
  DATATYPE H5T_STD_I64LE
  DATASPACE SIMPLE { ( <nodeCount> ) / ( <nodeCount> ) }
}

DATASET "position" {
COMMENT "The position of each node."
  DATATYPE H5T_IEEE_F64LE
  DATASPACE SIMPLE { ( <nodeCount>, 3 ) / ( <nodeCount>, 3 ) }
}

DATASET "scaleRadius" {
COMMENT "The scale radius of each node."
  DATATYPE H5T_IEEE_F64LE
  DATASPACE SIMPLE { ( <nodeCount> ) / ( <nodeCount> ) }
}

DATASET "redshift" {
COMMENT "The redshift of each node."
  DATATYPE H5T_IEEE_F64LE
  DATASPACE SIMPLE { ( <nodeCount> ) / ( <nodeCount> ) }
}

DATASET "spin" {
COMMENT "The spin of each node."
  DATATYPE H5T_IEEE_F64LE
  DATASPACE SIMPLE { ( <nodeCount>, 3 ) / ( <nodeCount>, 3 ) }
}

DATASET "time" {
COMMENT "The time of each node."
  DATATYPE H5T_IEEE_F64LE
  DATASPACE SIMPLE { ( <nodeCount> ) / ( <nodeCount> ) }
}

DATASET "velocity" {
COMMENT "The velocity of each node."
  DATATYPE H5T_IEEE_F64LE
  DATASPACE SIMPLE { ( <nodeCount>, 3 ) / ( <nodeCount>, 3 ) }
}

A.6.1. Standard Attributes

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;

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 §10.2 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;

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, λ, 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 §10.2 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;
A. Merger Tree File Format

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 position\(^1\) 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.

\(^1\)That 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.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:

```hdf5
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.

```hdf5
GROUP "particles" {
  DATASET "particleID" {
    COMMENT "The ID of each particle."
    DATATYPE H5T_STD_I64LE
    DATASPACE SIMPLE { ( <particleCount> ) / ( <particleCount> ) }
  }
  DATASET "redshift" {
    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
  }
}
```
A. Merger Tree File Format

\[
\text{DATASPACE SIMPLE} \{ \langle \text{particleCount}, 3 \rangle / \langle \text{particleCount}, 3 \rangle \} \\
\text{DATASET "velocity"} \{ \\
\text{COMMENT "The velocity of each node."} \\
\text{DATATYPE H5T_IEEE_F64LE} \\
\text{DATASPACE SIMPLE} \{ \langle \text{particleCount}, 3 \rangle / \langle \text{particleCount}, 3 \rangle \} \\
\}
\]

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 §10.2 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 merger trees that contain just the mass, redshift and descendent 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 descendent 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);
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:

```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:

```call mergerTrees%setParticleProperty(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:
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 metadata 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 §12.27.2). Export happens after any merger tree pre-evolution tasks (see §16.4.3).
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).\(^1\)

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 key left bottom\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')
   );
```

\(^1\)If 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...
B. Plotting Support

&PrettyPlots::Prepare_Dataset($plot, $x2Data, $y2Data, 
  errorDown => $errorDown, 
  errorUp => $errorUp, 
  title => "Galacticus", 
  style => point, 
  symbol => [6,7], 
  weight => [5,3], 
  color => $PrettyPlots::colorPairs{'redYellow'} 
);

&PrettyPlots::Plot_Datasets($gnuPlot,$plot);
close($gnuPlot);
&LaTeX::GnuPlot2PNG($outputFile.".eps", backgroundColor => "#000080", margin => 1);

The process begins by opening a pipe to GNUPlot and specifying the \texttt{epslatex} terminal along with \texttt{color} and \texttt{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 \textit{must} specify margins manually\footnote{The \texttt{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.}. Following this are calls to \texttt{&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 \textit{x} and \textit{y} data to be plotted. Following this are multiple options as follows:

\textbf{title} Gives the title of the dataset for inclusion in the plot key;

\textbf{style} Specifies how the dataset should be drawn: either \texttt{line}, \texttt{point}, \texttt{boxes}, or \texttt{filledCurve};

\textbf{linePattern} Specifies the line pattern (as defined for GNUPlot’s \texttt{lt} option) to use;

\textbf{symbol} A two element list giving the symbol indices that should be used to plot the border and inner parts of each point respectively;

\textbf{weight} A two element list giving the line weights to be used for border and inner parts of each point/line respectively;

\textbf{color} A two element list giving the color of border and inner parts of each point/line respectively. Colors should be specified as \texttt{#RRGGBB} in hexadecimal. Several suitable color pairs and sequences of pairs are defined in the \texttt{GnuPlot::PrettyPlots} module;

\textbf{pointSize} Specifies the size of the points to be used;

\textbf{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;

\textbf{filledCurve} If the \texttt{filledCurve} style is used, this option specifies the type of filled curve (\texttt{closed}, \texttt{x1}, \texttt{x2}, etc.—see the GNUPlot \texttt{help filledcurve} text for complete options). The default is \texttt{closed};

\textbf{y2} If the \texttt{filledCurve} style is used along with the \texttt{filledCurve=closed} option, this option is used to specify a second PDL of \textit{y}-axis values. The region between this curve and the usual \textit{y}-axis curve will be filled.
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\(^3\). 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 \texttt{dot} command, which is a part of \texttt{GraphViz} is useful for creating diagrams of merger trees. \texttt{Galacticus} provides a function to output the structure of any merger tree in \texttt{GraphViz} format. This function, \texttt{Merger_Tree_Dump}, is provided by the \texttt{Merger_Trees_Dump} module. Usage is as follows:

```plaintext
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 \texttt{index} is the tree index (successive calls to \texttt{Merger_Tree_Dump} with the same index will result in a sequence of output files—see below), and \texttt{baseNode} is a pointer to the base node of the tree to be dumped. All other arguments are optional:

- \texttt{highlightNodes} A list of node IDs. All nodes listed will be highlighted in the diagram;
- \texttt{backgroundColor} The color for the background of the diagram;
- \texttt{nodeColor} The color used to draw nodes;
- \texttt{highlightColor} The color used for highlighted nodes;
- \texttt{edgeColor} The color of edges (lines joining nodes);
- \texttt{nodeStyle} The style to use when drawing nodes;

\(^3\)This 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;

dgeStyle 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);

caleNodesByLogMass If true, the size of nodes will be set to be proportional to the logarithm of the node mass;

dgeLengthsToTimes 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


1403


1406


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. 151

**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). 1411

**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. 1409

**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. 474

**component** An individual physical system within a node, such as a dark matter halo, a galactic disk or a supermassive black hole. 8, 154, 157–161, 164–166, 168–176, 204, 206–209, 223, 234, 1409

**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. 1411

**forest** A collection of merger trees that are linked together by virtue of nodes which jump between trees. 121

**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. 235, 1409

**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. 111

**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). 255

**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. 136

**node** A single point in a merger tree, consisting of a dark matter halo and associated baryons. 120, 135, 136, 155, 157, 158, 160, 161, 167, 168, 171–175, 193, 196, 204, 216, 220, 230, 1076, 1113, 1296, 1409, 1410
**PAH**  Polycyclic aromatic hydrocarbons (PAH) are large organic molecules consisting of fused aromatic rings. 1411

**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 $t_1 > t_0$. 135, 136

**primary progenitor**  The progenitor of a given node which is regarded as the direct descendant of that node (often, but not always, the most massive progenitor). Other progenitors are considered to merge into this primary progenitor. 135, 136, 1409

**UUID**  A universally unique identifier—this is a label which uniquely identifies some object (in this case, a Galacticus model). 116
Acronyms

**ADAF** advection-dominated accretion flow. 180, 181, *Glossary: ADAF*

**CDM** cold dark matter. 3, 184

**CMB** cosmic microwave background. 33, 188, 478, 490

**DSL** domain-specific language. 8, *Glossary: DSL*

**IGM** intergalactic medium. 160, 179, 225, 461, 462, 561

**IMF** initial mass function. 58–60, 132, 211–213, 224, 225, 521, 524, 525, 559, 565, 577, 578, 1184–1186, 1195

**ISCO** innermost stable circular orbit. 180

**ISM** interstellar medium. 154, 227, 601, 602, 1185, 1186

**NFW** Navarro-Frenk-White (dark matter halo profile). 193, 593

**ODE** ordinary differential equation. 135, 453

**PAH** polycyclic aromatic hydrocarbon. 134, *Glossary: PAH*

**SED** spectral energy distribution. 115, 133, 134

**SNe** supernovae. 47, 48, 85, 87, 88, 132
Index

mergerTreeWeight, 112

accretion
  baryonic, 179
  black holes, 156, 158
  disk, 180
accretion disks, 180
active galactic nuclei (AGN)
  feedback, 154, 157
AGN, 111
analysis
  analysis
    galacticus, 115
    meta, 115
bar instability, 197
bias
  halo, 186
black holes
  accretion, 156, 158, 235
  jets, 235
  supermassive, 235
C (language) coding, 256
calculation reset task, 256
central galaxies
  identifying, 239
chemical state, 213
chemicals
  reaction rates, 220
clustering
  halo model, 112, 236
code
  directives, 449
coding
  mixed language, 256
components, 450
  creation, 455
  destruction, 455
  evolution, 456
  implementing, 451
  initialization, 455
  methods, 456
structure, 450
Condor, 21
configuration, 19
continuum radiation
  Lyman continuum, 111
cooling, 187, 235
  cooling function, 187
  freefall radius, 190
  infall radius, 190
  output, 235
  radius, 189, 235
  rate, 188, 235
  rate modifier, 189
  specific angular momentum, 190
  time, 191
  time available, 191
  time available for freefall, 192
cooling function, 187
cooling radius, 189
cooling time, 191
cosmology, 179
critical density
  mass scaling, 185
dark energy, 180
dark matter, 181
dark matter halo
  concentration, 123
  mass accretion history, 192, 496
  mass loss, 196, 501
  scale radius, 123
  spin, 124
  distribution, 197, 499
dark matter halos
  bias, 186
  mass function, 186
dark matter profile
  concentration, 194
  shape, 195
debugging
  OpenMP, 21
  restoring merger tree internal state, 216
density
INDEX

contrast, 235
critical, 184
virial, 185
density contrast, 235
density profile
  Einasto, 193, 195, 496
  hot halo, 209
  core radius, 210
  Navarro-Frenk-White, 193, 496
depdencies, computation, 255
descendent node, 235
directives, 449
disk
  star formation rate
    output, 164
disks
  stability, 197
dust
  attenuation, 132
  emission, 132
  extinction, 110
  reprocessing, 113
Einasto profile, 193, 195, 496
enumerations
  adjustElements, 258
  componentType, 258
  coordinateSystem, 258
  dataType, 258
  extrapolationType, 258
  massDistributionSymmetry, 259
  massType, 259
  metaDataType, 259
  metallicityScale, 259
  propertyType, 260
  rangeExpand, 260
  rangeExpandSignExpect, 260
  units, 261
  weightBy, 261
evolution, 456
  interrupt, 458
  main branch, 239
excursion sets, 27
feedback, 230
  AGN, 154, 157
  expulsive, 231, 551
filtering
  output, 583
filters
  broadband, 119
  flux
    sub-mm, 111
  freefall radius, 190
galactic structure, 203
  initial radii, 204
galacticusConfig.xml, 19
galaxies
  central, 239
  indices, 240
  satellite, 239
  tracing through outputs, 240
  weighting, 112
galaxy
  merging, 205
GRAPHViz, 1395
GRASIL, 113, 132
gravitational lensing, 112
half-light radius, 236
halo bias, 186
halo mass function, 186
halo model, 112, 236
halos
  host mass, 109
heating
  hot halo, 159
histograms, 117
history
  global, 105
hot halo
  density profile, 209
  core radius, 210
  heating, 159
  ram pressure force, 210
  ram pressure stripping, 210
  temperature profile, 211
hydrogen
  chemical, 220
indices
  galaxies, 240
  nodes, 240
infall radius, 190
initial mass function, 211, 212
  selection, 211
instantaneous recycling approximation, 132
intergalactic medium, 213, 525
interrupts, 458
labels, unique, 255
lensing
  gravitational, 112
lightcone, 130
lightcones, 236
linear growth, 184
load average, 27
Lyman continuum, 111
main branch
  evolution, 239
mass accretion history
  dark matter halo, 192, 496
mass loss
  dark matter halo, 196, 501
mass profile, 239
  half-light radius, 236
merger times, 124
merger tree
  fixed time steps, 219
  links, 239
  mass accretion history, 220
    monotonic, 219
N-body, 219
  pruning
    by hierarchy, 220
    by mass, 220
  regrid times, 219
  rendering, 240
  structure, 240
merger trees, 214
  branching, 217, 218
  building, 218, 1387
  exporting, 1391
  file formats, 1390
  graphing, 1395
  main branch, 239
N-body, 125
merging
  dynamical friction, 228
  galaxy, 205
  progenitor properties, 208
  remnant size, 206
  substructure, 228
metadata, 116
Millennium Simulation, 130
mixed language coding, 256
mock catalog, 130

N-body
  merger trees, 125
  Navarro-Frenk-White profile, 193, 496
  node
    descendent, 235
  nodes
    host mass, 109
    indices, 240
    isolated, 239
    substructure, 239
  numerical algorithms
    root finding, 253

OpenMP, 12
  debugging, 21
optimization, 256
orbits
  N-body, 127
  satellite
    outputting, 242
    setting, 127
    virial, 127
output
  filtering, 583
output groups, 584
outputs
  global history, 105
  rotation curve, 240
  star formation rate, 242
  velocity dispersion, 242
parallel, 12
parameters
  generating, 20
path
  GALACTICUS root, 20
plotting, 1393
Population III
  supernovae, 228
power spectrum
  non-linear, 183, 534
  outputting, 30
  primordial, 181
  variance, 182
  window function, 182
pre-derivative task, 584
radiation, 539
ram pressure force
  hot halo, 210
ram pressure stripping
INDEX

hot halo, 210
reaction rates
  chemical, 220
recycling
  instantaneous, 132
root finding, 253
rotation curve
  outputting, 240
run time
  tree construction, 115
  tree evolution, 115
samples
  volume limited, 112
satellite
  merger times, 124
  orbit
    outputting, 242
satellite galaxies
  identifying, 239
satellites
  host mass, 109
spheroid
  standard
    pseudo-angular momentum, 165
    radius, 165
  star formation rate
    output, 167
spin
  dark matter halo, 197, 499
star formation
  Blitz-Rosolowsky rule, 222
  extended Schmidt law, 222
  Kennicutt-Schmidt law, 221
  Krumholz-McKee-Tumlinson method, 222
  rate
    surface density, 221
    timescale, 223
star formation history, 587
  outputting, 588
  recording, 588
star formation rate
  output
    disk, 164
    spheroid, 167
    outputting, 242
    peak, 112
statistics
  histograms, 117
  Perl modules, 116
stellar populations, 224
structure formation, 181
substructure, 228
supermassive black holes
  binary separation, 231
  mergers, 234
  recoil velocity, 233
  separation growth rate, 232
supernovae
  feedback, 230, 231
  Population III, 228
  Type Ia, 228
task, calculation reset, 256
task, pre-derivative, 584
tasks
  output, 584
temperature profile
  hot halo, 211
timesteps
  criteria, 135
transfer function, 183
  warm dark matter, 184
unique ID, 256
unique labels, 255
units, 106
velocity dispersion
  outputting, 242
  virial, 243
warm dark matter
  transfer function, 184
a1 (function), 630
a2 (function), 630
abundance_pattern_lookup (function), 617
abundances (type), 885
abundances_add (function), 885
abundances_allocate_elemental_values (subroutine), 885
abundances_atomic_index (function), 886
abundances_builder (subroutine), 886
abundances_deserialize (subroutine), 886
abundances_destroy (subroutine), 886
abundances_divide (function), 886
abundances_dump (subroutine), 886
abundances_dump_raw (subroutine), 886
abundances_get_metallicity (function), 886
abundances_helium_mass_fraction (function), 886
abundances_helium_number_fraction (function), 887
abundances_hydrogen_mass_fraction (function), 887
abundances_hydrogen_number_fraction (function), 887
abundances_increment (subroutine), 887
abundances_initialize (subroutine), 887
abundances_is_zero (function), 887
abundances_mass_to_mass_fraction (subroutine), 887
abundances_mass_to_mass_fraction_packed (subroutine), 887
abundances_max (function), 887
abundances_multiply (function), 888
abundances_multiply_switched (function), 888
abundances_names (function), 888
abundances_output (subroutine), 888
abundances_output_count (subroutine), 888
abundances_output_names (subroutine), 888
abundances_property_count (function), 888
abundances_read_raw (subroutine), 888
abundances_reset (subroutine), 888
abundances_serialize (subroutine), 889
abundances_set_metallicity (subroutine), 889
abundances_set_to_unity (subroutine), 889
abundances_structure (module), 883
abundances_subtract (function), 889
accretion.Bondi_Hoyle_Lyttleton.F90 (file), 603
accretion.halo.F90 (file), 603
accretion.halo.null.F90 (file), 604
accretion.halo.simple.F90 (file), 606
accretion_disk_adaf_tabulate (subroutine), 608
accretion_disk_jet_power (function), 612
accretion_disk_jet_power_adaf (function), 608
accretion_disk_jet_power_eddington (function), 611
accretion_disk_jet_power_shakura_sunyaev (function), 612
accretion_disk_jet_power_switched (function), 613
accretion_disk_radiative_efficiency (function), 612
accretion_disk_radiative_efficiency_adaf (function), 608
accretion_disk_radiative_efficiency_eddington (function), 611
accretion_disk_radiative_efficiency_shakura_sunyaev (function), 613
accretion_disk_radiative_efficiency_switched (function), 613
accretion_disk_switched_adaf_fraction (function), 613
accretion_disk_switched_adaf_radiative_efficiency_scaling (function), 614
accretion_disks (module), 611
accretion_disks.ADAF.F90 (file), 608
accretion_disks.Eddington_limited.F90 (file), 610
accretion_disks.F90 (file), 611
accretion_disks.Shakura_Sunyaev.F90 (file), 612
accretion_disks.switched.F90 (file), 613
accretion_disks_adaf (module), 608
accretion_disks_adaf_get_parameters (subroutine), 608
accretion_disks_adaf_initialize (subroutine), 608
accretion_disks_eddington (module), 611
accretion_disks_eddington_initialize (subroutine), 611
accretion_disks.initialize (subroutine), 612
accretion_disks_shakura_sunyaev (module), 612
accretion_disks_shakura_sunyaev_initialize (subroutine), 613
accretion_disks_switched (module), 613
accretion_disks_switched_initialize (subroutine), 614
accretion_halos (module), 603
accretion_halos.initialize (subroutine), 603
accretion_halos_null (module), 605
accretion_halos_null_initialize (subroutine), 605
accretion_halos_simple (module), 606
accretion_halos_simple_initialize (subroutine), 606
adaf_alpha (function), 609
adaf_angular_momentum (function), 609
adaf_bh.jet.power (function), 609
adaf_disk.jet.power (function), 609
adaf_disk.jet.power_from_black_hole (function), 609
adaf_enthalpy (function), 609
adaf_enthalpy.angular_momentum.product (function), 609
adaf_field_enhancement (function), 609
adaf_fluid.angular.velocity (function), 609
adaf.gamma (function), 610
adaf.gamma.phi (function), 610
adaf.gamma_r (function), 610
adaf.height (function), 610
adaf.temperature (function), 610
adaf.v (function), 610
add_memory_component (subroutine), 1348
adiabatic_solver.mean.orbital.radius (function), 728
adiabatic_solver.mean.orbital.radius.derivative (function), 728
adjustl (interface), 812
adjustl_. (function), 812
adustr (interface), 812
adjustr_ (function), 812
agetabledes (function), 680
agetabledesdarkenergy (function), 676
alloc_array (interface), 1348
alloc_array_character_1d (subroutine), 1349
alloc_array_double_precision_1d (subroutine), 1349
alloc_array_double_precision_2d (subroutine), 1349
alloc_array_double_precision_3d (subroutine), 1349
alloc_array_double_precision_4d (subroutine), 1349
alloc_array_double_precision_5d (subroutine), 1349
alloc_array_integer_1d (subroutine), 1349
alloc_array_integer_2d (subroutine), 1350
alloc_array_integer_kind_int8_1d (subroutine), 1350
alloc_array_integer_kind_int8_2d (subroutine), 1350
alloc_array_logical_1d (subroutine), 1350
alloc_array_real_kind_quad_1d (subroutine), 1350
allocate_buffers (subroutine), 757
amaximumroot (function), 1273
angular_momentum_nfw_scale_free (function), 703
array_cumulate (interface), 1317
array_cumulate_double (function), 1317
array_index (interface), 1317
array_index_double (function), 1317
array_index_double_2d (function), 1318
array_index_integer (function), 1318
array_index_integer8 (function), 1318
array_is_monotonic (interface), 1318
array_is_monotonic_double (function), 1318
array_is_monotonic_integer8 (function), 1318
array_reverse (interface), 1318
array_reverse_double (function), 1318
array_reverse_real (function), 1318
array_utilities (module), 1317
array_which (subroutine), 1319
arrays_search (module), 881
assert (interface), 1359
assert_character_1d_array (subroutine), 1359
assert_character_scalar (subroutine), 1360
assert_double_1d_array (subroutine), 1360
assert_double_2d_array (subroutine), 1360
assert_double_3d_array (subroutine), 1360
assert_double_4d_array (subroutine), 1360
assert_double_5d_array (subroutine), 1360
assert_double_scalar (subroutine), 1360
assert_integer8_1d_array (subroutine), 1360
assert_integer8_scalar (subroutine), 1361
assert_integer_1d_array (subroutine), 1361
assert_integer_scalar (subroutine), 1361
assert_logical_1d_array (subroutine), 1361
assert_logical_scalar  (subroutine), 1361
assert_real_1d_array  (subroutine), 1361
assert_real_scalar  (subroutine), 1361
assert_varstring_1d_array (subroutine), 1361
assert_varstring_scalar (subroutine), 1361
assertresult  (type), 1362
assign_isolated_node_indices (subroutine), 840
assign_mergers (subroutine), 840
assign_scale_radii (subroutine), 840
assign_spin_parameters (subroutine), 840
assign_uniqueids_to_clones (subroutine), 840
assignment(=)  (interface), 812, 895, 923
atom_lookup  (function), 617
atomic.cross_sections.Compton.F90  (file), 614
atomic.cross_sections.ionization.photo.F90  (file), 614
atomic.cross_sections.ionization.photo.Verner.F90  (file), 615
atomic.data.F90  (file), 617
atomic.rates.ionization.collisional.F90  (file), 618
atomic.rates.ionization.collisional.Verner.F90  (file), 618
atomic.rates.recombination.radiative.F90  (file), 619
atomic.rates.recombination.radiative.Verner.F90  (file), 619
atomic_abundance  (function), 617
atomic_cross_section_compton  (function), 614
atomic_cross_section_ionization_photo  (function), 615
atomic_cross_section_ionization_photo_initialize (subroutine), 615
atomic_cross_section_ionization_photo_verner  (function), 615
atomic_cross_section_ionization_photo_verner_initialize (subroutine), 616
atomic_cross_sections_compton  (module), 614
atomic_cross_sections.ionization.photo  (module), 614
atomic_cross_sections.ionization.photo_verner  (module), 615
atomic_data  (module), 617
atomic_data_atoms_count  (function), 617
atomic_data_initialize  (subroutine), 617
atomic_mass  (function), 618
atomic_rate_ionization_collisional  (function), 618
atomic_rate_ionization_collisional_initialize (subroutine), 618
atomic_rate_ionization_collisional_verner  (function), 619
atomic_rate_ionization_collisional_verner_initialize (subroutine), 619
atomic_rate_recombination_radiative  (function), 619
atomic_rate_recombination_radiative_initialize (subroutine), 619
atomic_rate_recombination_radiative_verner  (function), 620
atomic_rate_recombination_radiative_verner_initialize (subroutine), 620
atomic_rates.ionization.collisional  (module), 618
atomic_rates.ionization.collisional_verner  (module), 618
atomic_rates.recombination.radiative  (module), 619
atomic_rates.recombination_radiative  (module), 620
atomic_short_label  (function), 618
atomicbond  (type), 893
atomicdata  (type), 618
atomicstructure (type), 894
b0 (function), 713
b1 (function), 714
bar_instability_timescale (function), 720
bar_instability_timescale_eln (function), 719
bar_instability_timescale_null (function), 720
basicaccretionrate (function), 923
basicaccretionrateisgettable (function), 923
basiccountlinked (function), 923
basiccreatebyinterrupt (subroutine), 923
basiccreatelinked (subroutine), 923
basicdestroylinked (subroutine), 923
basicget (function), 923
basicmass (function), 923
basicmassisgettable (function), 924
basicnonevolvingmassget (function), 924
basicnonevolvingmassset (subroutine), 924
basicnonevolvingtimecount (function), 924
basicnonevolvingtimeget (function), 924
basicnonevolvingtimelastisolatedset (subroutine), 924
basicnonevolvingtimerate (subroutine), 924
basicnonevolvingtimescale (subroutine), 924
basicnonevolvingtimeset (subroutine), 924
basicnullbindingdouble0inout (function), 925
basicnullbindinginteger0in (function), 925
basicnullbindingratedouble0inout (subroutine), 925
basicnullbindingrateinteger0in (subroutine), 925
basicnullbindingsetdouble0inout (subroutine), 925
basicnullbindingsetinteger0in (subroutine), 925
basicstandardaccretionrateset (subroutine), 925
basicstandardmasscount (function), 925
basicstandardmassget (function), 926
basicstandardmassrate (subroutine), 926
basicstandardmassscale (subroutine), 926
basicstandardmassset (subroutine), 926
basicstandardtimecount (function), 926
basicstandardtimeget (function), 926
basicstandardtimelastisolatedset (subroutine), 926
basicstandardtimerate (subroutine), 926
basicstandardtimescale (subroutine), 926
basicstandardtimeset (subroutine), 927
basic_time (function), 927
basic_timeisgettable (function), 927
basic_timeisgettable (function), 927
bessel_function_i0 (function), 823
bessel_function_i1 (function), 823
bessel_function_k0 (function), 823
bessel_function_k1 (function), 823
bessel_functions (module), 823
bivar (module), 620
bivar.F90 (file), 620
black_hole_binary_initial_radii (module), 621
black_hole_binary_initial_radii_spheroid_size (module), 622
black_hole_binary_initial_radii_spheroid_size_initialize (subroutine), 622
black_hole_binary_initial_radii_tidal_radius (module), 623
black_hole_binary_initial_radii_tidal_radius_initialize (subroutine), 623
black_hole_binary_initial_radii_volonteri_2003 (module), 622
black_hole_binary_initial_radii_volonteri_2003_initialize (subroutine), 622
black_hole_binary_initial_radius (function), 621
black_hole_binary_initial_radius_spheroid_size (function), 623
black_hole_binary_initial_radius_tidal_radius (function), 623
black_hole_binary_initial_radius_volonteri_2003 (function), 622
black_hole_binary_merger (subroutine), 628
black_hole_binary_merger_rezzolla (subroutine), 629
black_hole_binary_merger_rezzolla_initialize (subroutine), 629
black_hole_binary_mergers (module), 628
black_hole_binary_mergers_rezzolla (module), 628
black_hole_binary_recoil_velocities (module), 625
black_hole_binary_recoil_velocities_standard (module), 624
black_hole_binary_recoil_velocity (function), 625
black_hole_binary_recoil_velocity_null (function), 625
black_hole_binary_recoil_velocity_null_initialize (subroutine), 625
black_hole_binary_recoil_velocity_standard (function), 624
black_hole_binary_recoil_velocity_standard_initialize (subroutine), 624
black_hole_binary_recoil_velocity_standard_snapshot (subroutine), 624
black_hole_binary_recoil_velocity_standard_state_retrieve (subroutine), 624
black_hole_binary_recoil_velocity_standard_state_store (subroutine), 624
black_hole_binary_separation_growth_rate (function), 626
black_hole_binary_separation_growth_rate_null (function), 626
black_hole_binary_separation_growth_rate_null_initialize (subroutine), 626
black_hole_binary_separation_growth_rate_standard (function), 628
black_hole_binary_separation_growth_rate_standard_init (subroutine), 628
black_hole_binary_separations (module), 625
black_hole_binary_separations_null (module), 626
black_hole_binary_separations_standard (module), 626
black_hole_eddington_accretion_rate (function), 630
black_hole_frame_dragging_frequency (interface), 630
black_hole_frame_dragging_frequency_node (function), 630
black_hole_frame_dragging_frequency_spin (function), 630
black_hole_fundamentals (module), 629
black_hole_gravitational_radius (function), 630
black_hole_horizon_radius (interface), 630
black_hole_horizon_radius_node (function), 630
black_hole_horizon_radius_spin (function), 630
black_hole_isco_radius (interface), 631
black_hole_isco_radius_node (function), 631
black_hole_isco_radius_spin (function), 631
black_hole_isco_specific-angular_momentum (function), 631
black_hole_isco_specific-energy (interface), 631
black_hole_isco_specific-energy_node (function), 631
black_hole_isco_specific-energy-spin (function), 631
black_hole_metric_a_factor (interface), 631
black_hole_metric_a_factor_node (function), 632
black_hole_metric_a_factor-spin (function), 632
black_hole_metric_d_factor (interface), 632
black_hole_metric_d_factor_node (function), 632
black_hole_metric_d_factor-spin (function), 632
black_hole_recoil_velocities_null (module), 625
black_hole_rotational-energy-spin-down (interface), 632
black_hole_rotational-energy-spin-down_node (function), 632
black_hole_rotational-energy-spin-down-spin (function), 632
black_hole_spin-up_rate (function), 612
black_hole_spin-up_rate_adaf (function), 610
black_hole_spin-up_rate_eddington (function), 611
black_hole_spin-up_rate-shakura_sunyaev (function), 613
black_hole_static_radius (interface), 633
black_hole_static_radius_node (function), 634
black_hole_static_radius-spin (function), 634
black_holes.binaries.initial_radius.F90 (file), 621
black_holes.binaries.initial_radius.spheroid_size_fraction.F90 (file), 622
black_holes.binaries.initial_radius.tidal_radius.F90 (file), 623
black_holes.binaries.initial_radius.Volonteri_2003.F90 (file), 622
black_holes.binaries.recoil_velocity.Campanelli2007.F90 (file), 623
black_holes.binaries.recoil_velocity.F90 (file), 624
black_holes.binaries.recoil_velocity.null.F90 (file), 625
black_holes.binaries.separation_growth_rate.F90 (file), 625
black_holes.binaries.separation_growth_rate.null.F90 (file), 626
black_holes.binaries.separation_growth_rate.standard.F90 (file), 626
black_holes.binary_mergers.F90 (file), 628
black_holes.binary_mergers.Rezzolla2008.F90 (file), 628
black_holes.fundamentals.F90 (file), 628
blackbody_emission (function), 1303
blackholecountlinked (function), 927
blackholecreatebyinterrupt (subroutine), 927
blackholecreate_linked (subroutine), 927
blackholedeestroy_linked (subroutine), 927
blackholeget (function), 928
blackholemass (function), 928
blackholemassisgettable (function), 928
blackholemassseed (function), 928
blackholemassseedisgettable (function), 928
blackhole_null_binding_double0inout (function), 928
blackhole_null_binding_integer0in (function), 928
blackhole_null_binding_integerrad0inout (subroutine), 928
blackholenullbindingrateinteger0in (subroutine), 928
blackholenullbindingsetdouble0inout (subroutine), 928
blackholenullbindingsetinteger0in (subroutine), 929
blackholeradialposition (function), 929
blackholeradialpositionisgettable (function), 929
blackholesimplemasscount (function), 929
blackholesimplemassget (function), 929
blackholesimplemassrate (subroutine), 929
blackholesimplemassscale (subroutine), 929
blackholesimplemassset (subroutine), 929
blackholespin (function), 929
blackholespinisgettable (function), 930
blackholespinseed (function), 930
blackholespinseedisgettable (function), 930
blackholestandardmasscount (function), 930
blackholestandardmassget (function), 930
blackholestandardmassrate (subroutine), 930
blackholestandardmassscale (subroutine), 930
blackholestandardmassset (subroutine), 930
blackholestandardradialpositioncount (function), 930
blackholestandardradialpositionget (function), 931
blackholestandardradialpositionrate (subroutine), 931
blackholestandardradialpositionscale (subroutine), 931
blackholestandardradialpositionset (subroutine), 931
blackholestandardspincount (function), 931
blackholestandardspinrate (subroutine), 931
blackholestandardspinset (subroutine), 931
blackholestandardtripleinteractiontimeget (function), 931
blackholestandardtripleinteractiontimeset (subroutine), 932
blackholetripleinteractiontime (function), 932
blackholetripleinteractiontimeisgettable (function), 932
bondi_hoyle_lyttleton_accretion (module), 603
bondi_hoyle_lyttleton_accretion_radius (function), 603
bondi_hoyle_lyttleton_accretion_rate (function), 603
boolean_false (function), 932
boolean_true (function), 932
branching_probability_integrand (function), 831
branching_probability_integrand_generalized (function), 829
build_child_and_sibling_links (subroutine), 841
build_descendent_pointers (subroutine), 841
build_isolated_parent_pointers (subroutine), 841
build_parent_pointers (subroutine), 841
build_subhalo_mass_histories (subroutine), 841
c (function), 714
chabrier_phi (function), 1182
char (interface), 813
char_auto (function), 813
char_fixed (function), 813
chemical.reaction_rates.F90 (file), 634
chemical.reaction_rates.hydrogen.F90 (file), 634
chemical.reaction_rates.null.F90 (file), 639
chemical.reaction_rates.utilities.F90 (file), 640
chemical.state.atomic_CIE_cloudy.F90 (file), 645
chemical.state.CIE_file.F90 (file), 640
chemical.state.F90 (file), 643
chemical_abundances_add (function), 890
chemical_abundances_allocate_values (subroutine), 890
chemical_abundances_deserialize (subroutine), 891
chemical_abundances_divide (function), 891
chemical_abundances_increment (subroutine), 891
chemical_abundances_initialize (subroutine), 891
chemical_abundances_multiply (function), 891
chemical_abundances serialize (subroutine), 891
chemical_abundances_structure (module), 889
chemical_abundances_subtract (function), 891
chemical_database_get (subroutine), 894
chemical_database_get_index (function), 894
chemical_densities (subroutine), 643
chemical_densities_atomic_CIE_cloudy (subroutine), 645
chemical_densities_CIE_file (subroutine), 641
chemical_densities_CIE_file_interpolate (subroutine), 641
chemical_hydrogen_rate_h2_electron_to_2h_electron (subroutine), 635
chemical_hydrogen_rate_h2_gamma_to_2h (subroutine), 635
chemical_hydrogen_rate_h2_gamma_to_h2plus_electron (subroutine), 635
chemical_hydrogen_rate_h2_gamma_to_h2star_to_2h (subroutine), 635
chemical_hydrogen_rate_h2_h_to_3h (subroutine), 635
chemical_hydrogen_rate_h2_hplus_to_h2plus_h (subroutine), 635
chemical_hydrogen_rate_h2plus_electron_to_2h (subroutine), 636
chemical_hydrogen_rate_h2plus_gamma_to_2hplus_electron (subroutine), 636
chemical_hydrogen_rate_h2plus_gamma_to_h_hplus (subroutine), 636
chemical_hydrogen_rate_h2plus_h_to_h2_hplus (subroutine), 636
chemical_hydrogen_rate_h2plus_hminus_to_h2_h (subroutine), 636
chemical_hydrogen_rate_h_electron_to_hminus_photon (subroutine), 636
chemical_hydrogen_rate_h_electron_to_hplus_2electron (subroutine), 636
chemical_hydrogen_rate_h_minus_hgamma_to_2h_electron (subroutine), 636
chemical_hydrogen_rate_h_minus_hgamma_to_h_electron (subroutine), 637
chemical_hydrogen_rate_hplus_electron_to_h_hplus (subroutine), 637
chemical_hydrogen_rate_h_minus_hgamma_to_h_2electron (subroutine), 637
chemical_hydrogen_rate_h_minus_hgamma_to_h_electron (subroutine), 637
chemical_hydrogen_rate_hminus_hplus_to_2h (subroutine), 637
chemical_hydrogen_rate_hminus_hplus_to_h2plus_electron (subroutine), 637
chemical_hydrogen_rate_hplus_electron_to_h_photon (subroutine), 638
chemical_hydrogen_rates (module), 634
chemical_hydrogen_rates_compute (subroutine), 638
chemical_hydrogen_rates_initialize (subroutine), 638
chemical_reaction_rate (subroutine), 634
CODE INDEX

chemical_reaction_rates (module), 634
chemical_reaction_rates_initialize (subroutine), 634
chemical_reaction_rates_null (module), 639
chemical_reaction_rates_null_compute (subroutine), 640
chemical_reaction_rates_null_initialize (subroutine), 640
chemical_reaction_rates_utilities (module), 640
chemical_state_atomic_cie_cloudy_create (subroutine), 645
chemical_state_atomic_cie_cloudy_initialize (subroutine), 645
chemical_state_cie_chemicals_initialize (subroutine), 641
chemical_state_cie_file_format_version (function), 641
chemical_state_cie_file_initialize (subroutine), 641
chemical_state_cie_file_read (subroutine), 641
chemical_state_cie_file_read_initialize (subroutine), 642
chemical_state_initialize (subroutine), 643
chemical_states (module), 643
chemical_states_atomic_cie_cloudy (module), 645
chemical_states_cie_file (module), 640
chemical_structure_charge (function), 894
chemical_structure_export (subroutine), 894
chemical_structure_initialize (subroutine), 894
chemical_structure_mass (function), 894
chemical_structures (module), 893
chemical_abundances (type), 891
chemical_abundances (function), 891
chemical_abundances_destroy (subroutine), 892
chemical_abundances_reset (subroutine), 892
chemical_abundances_set (subroutine), 892
chemical_abundances_set_to_unity (subroutine), 892
chemicals_builder (subroutine), 892
chemicals_dump (subroutine), 892
chemicals_dump_raw (subroutine), 892
chemicals_enforce_positive (subroutine), 892
chemicals_index (function), 892
chemicals_mass_to_density_conversion (function), 640
chemicals_mass_to_number (subroutine), 893
chemicals_names (function), 893
chemicals_number_to_mass (subroutine), 893
chemicals_property_count (function), 893
chemicals_read_raw (subroutine), 893
chemicalstructure (type), 894
circularity_cumulative_probability (function), 1178
circularity_root (function), 1178
close_parameters_group (subroutine), 1331
cmb_temperature (function), 673
cmb_temperature_matter_dark_energy (function), 677
cmb_temperature_matter_lambda (function), 680
cmin (function), 714
cnode_component_basic_accretionrate (function), 1113
cnode_component_basic_mass (function), 1114
<table>
<thead>
<tr>
<th>Code Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>cnode_component_basic_time (function)</td>
<td>1114</td>
</tr>
<tr>
<td>cnode_component_basic_timelastisolated (function)</td>
<td>1114</td>
</tr>
<tr>
<td>cnode_component_blackhole_mass (function)</td>
<td>1114</td>
</tr>
<tr>
<td>cnode_component_blackhole_massseed (function)</td>
<td>1114</td>
</tr>
<tr>
<td>cnode_component_blackhole_radialposition (function)</td>
<td>1114</td>
</tr>
<tr>
<td>cnode_component_blackhole_spin (function)</td>
<td>1114</td>
</tr>
<tr>
<td>cnode_component_blackhole_spinseed (function)</td>
<td>1114</td>
</tr>
<tr>
<td>cnode_component_blackhole_tripleinteractiontime (function)</td>
<td>1114</td>
</tr>
<tr>
<td>cnode_component_darkmatterprofile_scale (function)</td>
<td>1114</td>
</tr>
<tr>
<td>cnode_component_darkmatterprofile_scalegrowthrate (function)</td>
<td>1115</td>
</tr>
<tr>
<td>cnode_component_darkmatterprofile_shape (function)</td>
<td>1115</td>
</tr>
<tr>
<td>cnode_component_darkmatterprofile_shapegrowthrate (function)</td>
<td>1115</td>
</tr>
<tr>
<td>cnode_component_disk_angularmomentum (function)</td>
<td>1115</td>
</tr>
<tr>
<td>cnode_component_disk_halfmassradius (function)</td>
<td>1115</td>
</tr>
<tr>
<td>cnode_component_disk_massgas (function)</td>
<td>1115</td>
</tr>
<tr>
<td>cnode_component_disk_massstellar (function)</td>
<td>1115</td>
</tr>
<tr>
<td>cnode_component_disk_radius (function)</td>
<td>1115</td>
</tr>
<tr>
<td>cnode_component_disk_starformationrate (function)</td>
<td>1115</td>
</tr>
<tr>
<td>cnode_component_disk_velocity (function)</td>
<td>1115</td>
</tr>
<tr>
<td>cnode_component_formationtime_formationtime (function)</td>
<td>1116</td>
</tr>
<tr>
<td>cnode_component_hothalo_angularmomentum (function)</td>
<td>1116</td>
</tr>
<tr>
<td>cnode_component_hothalo_mass (function)</td>
<td>1116</td>
</tr>
<tr>
<td>cnode_component_hothalo_outerradius (function)</td>
<td>1116</td>
</tr>
<tr>
<td>cnode_component_hothalo_outflowedangularmomentum (function)</td>
<td>1116</td>
</tr>
<tr>
<td>cnode_component_hothalo_outflowedmass (function)</td>
<td>1116</td>
</tr>
<tr>
<td>cnode_component_hothalo_strippedmass (function)</td>
<td>1116</td>
</tr>
<tr>
<td>cnode_component_hothalo_unaccretedmass (function)</td>
<td>1116</td>
</tr>
<tr>
<td>cnode_component_interoutput_diskstarformationrate (function)</td>
<td>1116</td>
</tr>
<tr>
<td>cnode_component_interoutput_spheroidstarformationrate (function)</td>
<td>1117</td>
</tr>
<tr>
<td>cnode_component_mergingstatistics_galaxymajormergertime (function)</td>
<td>1117</td>
</tr>
<tr>
<td>cnode_component_mergingstatistics_nodeformationtime (function)</td>
<td>1117</td>
</tr>
<tr>
<td>cnode_component_mergingstatistics_nodemajormergertime (function)</td>
<td>1117</td>
</tr>
<tr>
<td>cnode_component_satellite_boundmass (function)</td>
<td>1117</td>
</tr>
<tr>
<td>cnode_component_satellite_merge time (function)</td>
<td>1117</td>
</tr>
<tr>
<td>cnode_component_satellite_timeofmerging (function)</td>
<td>1117</td>
</tr>
<tr>
<td>cnode_component_spheroid_angularmomentum (function)</td>
<td>1117</td>
</tr>
<tr>
<td>cnode_component_spheroid_halfmassradius (function)</td>
<td>1117</td>
</tr>
<tr>
<td>cnode_component_spheroid_massgas (function)</td>
<td>1117</td>
</tr>
<tr>
<td>cnode_component_spheroid_massstellar (function)</td>
<td>1118</td>
</tr>
<tr>
<td>cnode_component_spheroid_radius (function)</td>
<td>1118</td>
</tr>
<tr>
<td>cnode_component_spheroid_starformationrate (function)</td>
<td>1118</td>
</tr>
<tr>
<td>cnode_component_spheroid_velocity (function)</td>
<td>1118</td>
</tr>
<tr>
<td>cnode_component_spingrowthrate (function)</td>
<td>1118</td>
</tr>
<tr>
<td>code_memory_usage (subroutine)</td>
<td>1350</td>
</tr>
<tr>
<td>collapsedef (function)</td>
<td>680</td>
</tr>
<tr>
<td>collapsedef (function)</td>
<td>680</td>
</tr>
<tr>
<td>collapsing_mass_root (function)</td>
<td>1239</td>
</tr>
<tr>
<td>command_arguments (module)</td>
<td>1319</td>
</tr>
</tbody>
</table>
comoving_distance (function), 673
comoving_distance_conversion (function), 674
comoving_distance_conversion_matter_dark_energy (function), 677
comoving_distance_conversion_matter_lambda (function), 680
comoving_distance_integrand (function), 680
comoving_distance_matter_dark_energy (function), 677
comoving_distance_matter_lambda (function), 680
comoving_volume_element_redshift (function), 674
comoving_volume_element_time (function), 674
compare_double (function), 882
compare_integer (function), 882
compare_integer8 (function), 882
component_density (function), 721
component_enclosed_mass (function), 722, 728
component_potential (function), 725
component_rotation_curve (function), 728, 732
component_rotation_curve_gradient (function), 728, 733
component_surface_density (function), 733
compute_common_factors (subroutine), 829, 831
conditional_varstr_integer (function), 1357
conditional_varstr_integer8 (function), 1357
conditional_stellar_mass_functions (module), 784
conditional_stellar_mass_functions_behroozi2010 (module), 783
conditional_stellar_mass_functions_behroozi2010_initialize (subroutine), 783
conditional_stellar_mass_functions_initialize (subroutine), 784
constant.F90 (file), 646
constants_nswc (module), 646
convert_varstring_to_char (function), 1357
cooling.cooling_function.atomic_CIE_Cloudy.F90 (file), 651
cooling.cooling_function.CIE_file.F90 (file), 646
cooling.cooling_function.CMB_Compton.F90 (file), 648
cooling.cooling_function.F90 (file), 649
cooling.cooling_function.molecular_hydrogen_Galli_Palla.F90 (file), 652
cooling.cooling_radius.F90 (file), 654
cooling.cooling_radius.isothermal_profile.F90 (file), 655
cooling.cooling_radius.simple.F90 (file), 656
cooling.cooling_rate.F90 (file), 658
cooling.cooling_rate.modifier.cut_off.F90 (file), 659
cooling.cooling_rate.simple.F90 (file), 660
cooling.cooling_rate.simple_scaling.F90 (file), 660
cooling.cooling_rate.White-Frenk.F90 (file), 659
cooling.cooling_rate.zero.F90 (file), 661
cooling.cooling_time.F90 (file), 661
cooling.cooling_time.simple.F90 (file), 662
cooling.freefall_radii.dark_matter_halo.F90 (file), 663
cooling.freefall_radii.F90 (file), 663
cooling.freefall_time_available.F90 (file), 664
cooling.freefall_time_available.halo_formation.F90 (file), 665
cooling.infall_radius.cooling_and_freefall.F90 (file), 666
cooling.infall_radius.cooling_radius.F90 (file), 667
cooling.infall_radius.F90 (file), 666
cooling.specific_angular_momentum.constant_rotation.F90 (file), 668
cooling.specific_angular_momentum.F90 (file), 667
cooling.specific_angular_momentum.mean.F90 (file), 669
cooling.time_available.F90 (file), 669
cooling.time_available.halo_formation.F90 (file), 671
cooling.time_available.White-Frenk.F90 (file), 670
cooling_freefall_time_available (function), 664
cooling_freefall_time_available_get_template (function), 665
cooling_freefall_time_available_get_template (interface), 664
cooling_freefall_time_available_increase_rate (function), 665
cooling_freefall_time_available_initialize (subroutine), 665
cooling_freefall_times_available (module), 664
cooling_function (function), 649
cooling_function_atomic_cie_cloudy (subroutine), 652
cooling_function_atomic_cie_cloudy_create (subroutine), 652
cooling_function_atomic_cie_cloudy_initialize (subroutine), 652
cooling_function_cie_file (subroutine), 647
cooling_function_cie_file_format_version (function), 647
cooling_function_cie_file_initialize (subroutine), 647
cooling_function_cie_file_interpolate (function), 647
cooling_function_cie_file_logtemperature_interpolate (function), 647
cooling_function_cie_file_read (subroutine), 647
cooling_function_cie_file_read_initialize (subroutine), 648
cooling_function_cmb_compton (subroutine), 649
cooling_function_cmb_compton_initialize (subroutine), 649
cooling_function_density_log_slope (function), 651
cooling_function_density_slope_atomic_cie_cloudy (subroutine), 652
cooling_function_density_slope_cie_file (subroutine), 648
cooling_function_density_slope_cmb_compton (subroutine), 649
cooling_function_density_slope_molecular_hydrogen_gp (subroutine), 653
cooling_function_gp_h2plus_electron (function), 653
cooling_function_gp_h_h2 (function), 653
cooling_function_gp_h_h2plus (function), 653
cooling_function_initialize (subroutine), 651
cooling_function_molecular_hydrogen_gp (subroutine), 653
cooling_function_molecular_hydrogen_gp_initialize (subroutine), 653
cooling_function_not_matched (subroutine), 651
cooling_function_temperature_log_slope (function), 651
cooling_function_temperature_slope_atomic_cie_cloudy (subroutine), 652
cooling_function_temperature_slope_cie_file (subroutine), 648
cooling_function_temperature_slope_cmb_compton (subroutine), 649
cooling_function_temperature_slope_molecular_hydrogen_gp (subroutine), 654
cooling_functions (module), 649
cooling_functions_atomic_cie_cloudy (module), 651
cooling_functions_cie_file (module), 647
cooling_functions_cmb_compton (module), 648
cooling_functions_molecular_hydrogen_galli_palla (module), 652
cooling_infall_radii (module), 666
cooling_radii (module), 654
cooling_radii_isothermal (module), 655
cooling_radii_simple (module), 656
cooling_radius (function), 654
cooling_radius_growth_rate (function), 654
cooling_radius_growth_rate_isothermal (function), 655
cooling_radius_growth_rate_simple (function), 656
cooling_radius_hot_halo_output (subroutine), 654
cooling_radius_hot_halo_output_count (subroutine), 655
cooling_radius_hot_halo_output_names (subroutine), 655
cooling_radius_initialize (subroutine), 655
cooling_radius_isothermal (function), 656
cooling_radius_isothermal_initialize (subroutine), 656
cooling_radius_isothermal_reset (subroutine), 656
cooling_radius_output_initialize (subroutine), 655
cooling_radius_root (function), 656
cooling_radius_simple (function), 657
cooling_radius_simple_initialize (subroutine), 657
cooling_radius_simple_reset (subroutine), 657
cooling_radius_solver_initialize (subroutine), 657
cooling_rate (function), 658
cooling_rate_cole2000 (function), 657
cooling_rate_cole2000_initialize (subroutine), 657
cooling_rate_hot_halo_output (subroutine), 658
cooling_rate_hot_halo_output_count (subroutine), 658
cooling_rate_hot_halo_output_names (subroutine), 658
cooling_rate_initialize (subroutine), 658
cooling_rate_modifier_cut_off (subroutine), 659
cooling_rate_output_initialize (subroutine), 659
cooling_rate_simple (function), 660
cooling_rate_simple_initialize (subroutine), 660
cooling_rate_simple_scaling (function), 660
cooling_rate_simple_scaling_initialize (subroutine), 661
cooling_rate_white_frenk (function), 659
cooling_rate_white_frenk_initialize (subroutine), 659
cooling_rate_zero (function), 661
cooling_rate_zero_initialize (subroutine), 661
cooling_rates (module), 658
cooling_rates_cole2000 (module), 657
cooling_rates_modifier_cut_off (module), 659
cooling_rates_simple (module), 660
cooling_rates_simple_scaling (module), 660
cooling_rates_white_frenk (module), 659
cooling_rates_zero (module), 661
cooling_specific_am_constant_rotation_initialize (subroutine), 668
cooling_specific_am_constant_rotation_reset (subroutine), 668
cooling_specific_am_mean_initialize (subroutine), 669
<table>
<thead>
<tr>
<th>Function/Module</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>cooling_specific_angular_momenta (module)</td>
<td>667</td>
</tr>
<tr>
<td>cooling_specific_angular_momenta_constant_rotation (module)</td>
<td>668</td>
</tr>
<tr>
<td>cooling_specific_angular_momenta_mean (module)</td>
<td>669</td>
</tr>
<tr>
<td>cooling_specific_angular_momentum (function)</td>
<td>668</td>
</tr>
<tr>
<td>cooling_specific_angular_momentum_constant_rotation (function)</td>
<td>668</td>
</tr>
<tr>
<td>cooling_specific_angular_momentum_initialize (subroutine)</td>
<td>668</td>
</tr>
<tr>
<td>cooling_specific_angular_momentum_mean (function)</td>
<td>669</td>
</tr>
<tr>
<td>cooling_time (function)</td>
<td>661</td>
</tr>
<tr>
<td>cooling_time_available (function)</td>
<td>669</td>
</tr>
<tr>
<td>cooling_time_available_get_template (function)</td>
<td>670</td>
</tr>
<tr>
<td>cooling_time_available_halo_formation (function)</td>
<td>671</td>
</tr>
<tr>
<td>cooling_time_available_halo_formation_initialize (subroutine)</td>
<td>671</td>
</tr>
<tr>
<td>cooling_time_available_increase_rate (function)</td>
<td>670</td>
</tr>
<tr>
<td>cooling_time_available_increase_rate_halo_formation (function)</td>
<td>671</td>
</tr>
<tr>
<td>cooling_time_available_increase_rate_wf (function)</td>
<td>670</td>
</tr>
<tr>
<td>cooling_time_available_initialize (subroutine)</td>
<td>670</td>
</tr>
<tr>
<td>cooling_time_available_wf (function)</td>
<td>670</td>
</tr>
<tr>
<td>cooling_time_available_wf_initialize (subroutine)</td>
<td>670</td>
</tr>
<tr>
<td>cooling_time_available_white_frenk (module)</td>
<td>670</td>
</tr>
<tr>
<td>cooling_time_density_log_slope (function)</td>
<td>662</td>
</tr>
<tr>
<td>cooling_time_density_log_slope_simple (function)</td>
<td>662</td>
</tr>
<tr>
<td>cooling_time_initialize (subroutine)</td>
<td>662</td>
</tr>
<tr>
<td>cooling_time_simple (function)</td>
<td>662</td>
</tr>
<tr>
<td>cooling_time_temperature_log_slope (function)</td>
<td>662</td>
</tr>
<tr>
<td>cooling_time_temperature_log_slope_simple (function)</td>
<td>663</td>
</tr>
<tr>
<td>cooling_times (module)</td>
<td>661</td>
</tr>
<tr>
<td>cooling_times_available (module)</td>
<td>669</td>
</tr>
<tr>
<td>cooling_times_available_halo_formation (module)</td>
<td>671</td>
</tr>
<tr>
<td>coordinate (type)</td>
<td>895</td>
</tr>
<tr>
<td>coordinate_systems (module)</td>
<td>781</td>
</tr>
<tr>
<td>coordinate_cartesian (type)</td>
<td>895</td>
</tr>
<tr>
<td>coordinate_cylindrical (type)</td>
<td>895</td>
</tr>
<tr>
<td>coordinates (module)</td>
<td>895</td>
</tr>
<tr>
<td>coordinates_assign (subroutine)</td>
<td>895</td>
</tr>
<tr>
<td>coordinates_assign_from (subroutine)</td>
<td>895</td>
</tr>
<tr>
<td>coordinates_assign_to (subroutine)</td>
<td>895</td>
</tr>
<tr>
<td>coordinates_cartesian_from_cartesian (subroutine)</td>
<td>895</td>
</tr>
<tr>
<td>coordinates_cartesian_set_x (subroutine)</td>
<td>896</td>
</tr>
<tr>
<td>coordinates_cartesian_set_y (subroutine)</td>
<td>896</td>
</tr>
<tr>
<td>coordinates_cartesian_set_z (subroutine)</td>
<td>896</td>
</tr>
<tr>
<td>coordinates_cartesian_to_cartesian (function)</td>
<td>896</td>
</tr>
<tr>
<td>coordinates_cartesian_to_cylindrical (function)</td>
<td>782</td>
</tr>
<tr>
<td>coordinates_cartesian_to_spherical (function)</td>
<td>782</td>
</tr>
<tr>
<td>coordinates_cartesian_x (function)</td>
<td>896</td>
</tr>
<tr>
<td>coordinates_cartesian_y (function)</td>
<td>896</td>
</tr>
<tr>
<td>coordinates_cartesian_z (function)</td>
<td>896</td>
</tr>
</tbody>
</table>
coordinates_cylindrical_from_cartesian (subroutine), 896
coordinates_cylindrical_phi (function), 896
coordinates_cylindrical_r (function), 896
coordinates_cylindrical_set_phi (subroutine), 897
coordinates_cylindrical_set_r (subroutine), 897
coordinates_cylindrical_set_z (subroutine), 897
coordinates_cylindrical_to_cartesian (function), 897
coordinates_cylindrical_to_spherical (function), 782
coordinates_cylindrical_z (function), 897
coordinates_null_from (subroutine), 897
coordinates_null_to (function), 897
coordinates_spherical_from_cartesian (subroutine), 897
coordinates_spherical_phi (function), 897
coordinates_spherical_r (function), 898
coordinates_spherical_set_phi (subroutine), 898
coordinates_spherical_set_r (subroutine), 898
coordinates_spherical_set_theta (subroutine), 898
coordinates_spherical_theta (function), 898
cosmetic_integral (function), 824
cosmic_time_is_valid (function), 674
cosmic_time_is_valid_matter_dark_energy (function), 677
cosmic_time_is_valid_matter_lambda (function), 680
cosmological_mass_root_variance (function), 1262
cosmological_mass_root_variance_logarithmic_derivative (function), 1262
cosmological_mass_root_variance_plus_logarithmic_derivative (subroutine), 1262
cosmological_mass_variance_filtered_power_spectrum (module), 1237
cosmological_mass_variance_filtered_power_spectrum_initialize (subroutine), 1237
cosmological_mass_variance_filtered_power_spectrum_tabulate (subroutine), 1237
cosmological_mass_variance_fps_state_retrieve (subroutine), 1238
cosmological_mass_variance_fps_state_store (subroutine), 1238
cosmological_parameters (module), 683
cosmology.functions.F90 (file), 671
cosmology.functions.matter_dark_energy.F90 (file), 676
cosmology.functions.matter_lambda.F90 (file), 679
cosmology.functions.options.F90 (file), 683
cosmology.functions.parameters.F90 (file), 683
cosmology.parameters.F90 (file), 683
cosmology_age (function), 674
cosmology_age_matter_dark_energy (function), 677
cosmology_age_matter_lambda (function), 681
cosmology_dark_energy_equation_of_state (function), 674
cosmology_dark_energy_equation_of_state_dark_energy (function), 677
cosmology_dark_energy_equation_of_state_dark_energy (function), 677
cosmology_dark_energy_equation_of_state_matter_lambda (function), 681
cosmology_dark_energy_exponent (function), 674
cosmology_dark_energy_exponent_dark_energy (function), 677
cosmology_dark_energy_exponent_matter_lambda (function), 681
cosmology_functions (module), 671
cosmology_functions_initialize (subroutine), 674
cosmology_functions_matter_dark_energy (module), 676
cosmology_functions_matter_dark_energy_initialize (subroutine), 677
cosmology_functions_matter_lambda (module), 679
cosmology_functions_matter_lambda_initialize (subroutine), 681
cosmology_functions_options (module), 683
cosmology_functions_parameters (module), 683
cosmology_matter_dark_energy_state_retrieve (subroutine), 677
cosmology_matter_dark_energy_state_store (subroutine), 678
cosmology_matter_lambda_state_retrieve (subroutine), 681
cosmology_matter_lambda_state_store (subroutine), 681
count_lines_in_file (interface), 1321
count_lines_in_file_char (function), 1321
count_lines_in_file_varstr (function), 1321
count_properties (subroutine), 757
create_branch_jump_event (subroutine), 841
create_indentation_format (subroutine), 738
create_node_array (subroutine), 841
create_node_indices (subroutine), 841
critical_density (function), 685
critical_overdensities_kitayama_suto1996 (module), 1240
critical_overdensity (module), 1238
critical_overdensity_collapsing_mass (function), 1239
critical_overdensity_for-collapse (function), 1239
critical_overdensity_for-collapse_time_gradient (function), 1239
critical_overdensity_kitayama_suto1996 (subroutine), 1241
critical_overdensity_kitayama_suto1996_initialize (subroutine), 1241
critical_overdensity_mass_scaling (function), 1240
critical_overdensity_mass_scaling_gradient (function), 1240
critical_overdensity_mass_scaling_gradient_null (function), 1241
critical_overdensity_mass_scaling_gradient_wdm (function), 1242
critical_overdensity_mass_scaling_initialize (subroutine), 1240
critical_overdensity_mass_scaling_null (function), 1241
critical_overdensity_mass_scaling_wdm (function), 1242
critical_overdensity_mass_scaling_wdm_initialize (subroutine), 1242
critical_overdensity_mass_scalings_null (module), 1241
critical_overdensity_mass_scalings_wdm (module), 1242
critical_overdensity_state_retrieve (subroutine), 1240
cross_section_h2_gamma_to_2h (function), 638
cross_section_h2_gamma_to_h2plus_electron (function), 638
cross_section_h2plus_gamma_to_2hplus_electron (function), 638
cross_section_h2plus_gamma_to_h_hplus (function), 638
cross_section_h_gamma_to_hplus_electron (function), 639
cross_section_hminus_gamma_to_h_electron (function), 639
cross_section_integrand (function), 1150
cumulative_conditional_stellar_mass_function (function), 784
<table>
<thead>
<tr>
<th>Function/Module/Initialization</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>dark_matter_halo_virial_radius_growth_rate (function)</td>
<td>693</td>
</tr>
<tr>
<td>dark_matter_halo_virial_temperature (function)</td>
<td>693</td>
</tr>
<tr>
<td>dark_matter_halo_virial_velocity (function)</td>
<td>693</td>
</tr>
<tr>
<td>dark_matter_halo_virial_velocity_growth_rate (function)</td>
<td>693</td>
</tr>
<tr>
<td>dark_matter_halos.formation_times.F90 (file)</td>
<td>686</td>
</tr>
<tr>
<td>dark_matter_halos.mass_accretion_history.F90 (file)</td>
<td>686</td>
</tr>
<tr>
<td>dark_matter_halos.mass_accretion_history.Wechsler2002.F90 (file)</td>
<td>687</td>
</tr>
<tr>
<td>dark_matter_halos.mass_accretion_history.Zhao2009.F90 (file)</td>
<td>687</td>
</tr>
<tr>
<td>dark_matter_halos.mass_loss_rates.F90 (file)</td>
<td>688</td>
</tr>
<tr>
<td>dark_matter_halos.mass_loss_rates.null.F90 (file)</td>
<td>689</td>
</tr>
<tr>
<td>dark_matter_halos.mass_loss_rates.vandenbosch.F90 (file)</td>
<td>689</td>
</tr>
<tr>
<td>dark_matter_halos.scales.F90 (file)</td>
<td>690</td>
</tr>
<tr>
<td>dark_matter_halos.spins.distributions.delta_function.F90 (file)</td>
<td>695</td>
</tr>
<tr>
<td>dark_matter_halos.spins.distributions.F90 (file)</td>
<td>695</td>
</tr>
<tr>
<td>dark_matter_halos.spins.distributions.lognormal.F90 (file)</td>
<td>696</td>
</tr>
<tr>
<td>dark_matter_halos.spins.F90 (file)</td>
<td>693</td>
</tr>
<tr>
<td>dark_matter_halos_mass_loss_rate (function)</td>
<td>688</td>
</tr>
<tr>
<td>dark_matter_halos_mass_loss_rate_null (function)</td>
<td>689</td>
</tr>
<tr>
<td>dark_matter_halos_mass_loss_rate_null_initialize (subroutine)</td>
<td>689</td>
</tr>
<tr>
<td>dark_matter_halos_mass_loss_rate_vandenbosch (function)</td>
<td>689</td>
</tr>
<tr>
<td>dark_matter_halos_mass_loss_rate_vandenbosch_initialize (subroutine)</td>
<td>689</td>
</tr>
<tr>
<td>dark_matter_halos_mass_loss_rates (module)</td>
<td>688</td>
</tr>
<tr>
<td>dark_matter_halos_mass_loss_rates_null (module)</td>
<td>689</td>
</tr>
<tr>
<td>dark_matter_halos_mass_loss_rates_vandenbosch (module)</td>
<td>689</td>
</tr>
<tr>
<td>dark_matter_mass_accretion_initialize (subroutine)</td>
<td>687</td>
</tr>
<tr>
<td>dark_matter_mass_accretion_wechsler2002_initialize (subroutine)</td>
<td>687</td>
</tr>
<tr>
<td>dark_matter_mass_accretion_zhao2009_initialize (subroutine)</td>
<td>688</td>
</tr>
<tr>
<td>dark_matter_profile_circular_velocity (function)</td>
<td>701</td>
</tr>
<tr>
<td>dark_matter_profile_circular_velocity_einasto (function)</td>
<td>697</td>
</tr>
<tr>
<td>dark_matter_profile_circular_velocity_isothermal (function)</td>
<td>709</td>
</tr>
<tr>
<td>dark_matter_profile_circular_velocity_nfw (function)</td>
<td>703</td>
</tr>
<tr>
<td>dark_matter_profile_concentration (function)</td>
<td>711</td>
</tr>
<tr>
<td>dark_matter_profile_concentration_gao2008 (function)</td>
<td>712</td>
</tr>
<tr>
<td>dark_matter_profile_concentration_munozcuartas2011 (function)</td>
<td>712</td>
</tr>
<tr>
<td>dark_matter_profile_concentration_nfw1996 (function)</td>
<td>713</td>
</tr>
<tr>
<td>dark_matter_profile_concentration_prada2011 (function)</td>
<td>714</td>
</tr>
<tr>
<td>dark_matter_profile_concentration_zhao2009 (function)</td>
<td>715</td>
</tr>
<tr>
<td>dark_matter_profile_density (function)</td>
<td>701</td>
</tr>
<tr>
<td>dark_matter_profile_density_einasto (function)</td>
<td>697</td>
</tr>
<tr>
<td>dark_matter_profile_density_isothermal (function)</td>
<td>709</td>
</tr>
<tr>
<td>dark_matter_profile_density_nfw (function)</td>
<td>703</td>
</tr>
<tr>
<td>dark_matter_profile_density_task (function)</td>
<td>716</td>
</tr>
<tr>
<td>dark_matter_profile_einasto_freefall_tabulate (subroutine)</td>
<td>697</td>
</tr>
<tr>
<td>dark_matter_profile_einasto_initialize (subroutine)</td>
<td>697</td>
</tr>
<tr>
<td>dark_matter_profile_enclosed_mass (function)</td>
<td>702</td>
</tr>
<tr>
<td>dark_matter_profile_enclosed_mass_einasto (function)</td>
<td>697</td>
</tr>
<tr>
<td>dark_matter_profile_enclosed_mass_isothermal (function)</td>
<td>709</td>
</tr>
<tr>
<td>dark_matter_profile_enclosed_mass_nfw (function)</td>
<td>704</td>
</tr>
</tbody>
</table>
dark_matter_profile_enclosed_mass_task (function), 716
dark_matter_profile_energy (function), 702
dark_matter_profile_energy_einasto (function), 698
dark_matter_profile_energy_growth_rate (function), 702
dark_matter_profile_energy_growth_rate_einasto (function), 698
dark_matter_profile_energy_growth_rate_isothermal (function), 709
dark_matter_profile_energy_growth_rate_nfw (function), 704
dark_matter_profile_energy_isothermal (function), 709
dark_matter_profile_energy_nfw (function), 704
dark_matter_profile_freefall_radius (function), 702
dark_matter_profile_freefall_radius_einasto (function), 698
dark_matter_profile_freefall_radius_increase_rate (function), 702
dark_matter_profile_freefall_radius_increase_rate_einasto (function), 698
dark_matter_profile_freefall_radius_increase_rate_isothermal (function), 710
dark_matter_profile_freefall_radius_increase_rate_nfw (function), 704
dark_matter_profile_freefall_radius_isothermal (function), 710
dark_matter_profile_freefall_radius_nfw (function), 704
dark_matter_profile_initialize (subroutine), 702
dark_matter_profile_isothermal_initialize (subroutine), 710
dark_matter_profile_kspace (function), 702
dark_matter_profile_kspace_einasto (function), 698
dark_matter_profile_kspace_isothermal (function), 710
dark_matter_profile_kspace_nfw (function), 704
dark_matter_profile_nfw_freefall_tabulate (subroutine), 704
dark_matter_profile_nfw_initialize (subroutine), 705
dark_matter_profile_nfw_inverse_angular_momentum (subroutine), 705
dark_matter_profile_nfw_reset (subroutine), 705
dark_matter_profile_nfw_tabulate (subroutine), 705
dark_matter_profile_potential (function), 702
dark_matter_profile_potential_einasto (function), 698
dark_matter_profile_potential_isothermal (function), 710
dark_matter_profile_potential_nfw (function), 705
dark_matter_profile_potential_task (function), 717
dark_matter_profile_radius_from_specific_angular_momentum (function), 702
dark_matter_profile_rotation_curvegradient_task (function), 717
dark_matter_profile_rotation_curve_task (function), 717
dark_matter_profile_rotation_normalization (function), 703
dark_matter_profile_rotation_normalization_einasto (function), 698
dark_matter_profile_rotation_normalization_isothermal (function), 711
dark_matter_profile_rotation_normalization_nfw (function), 705
dark_matter_profile_shape (function), 715
dark_matter_profile_shape_gao2008 (function), 716
dark_matter_profile_structure_tasks (module), 716
dark_matter_profiles (module), 701
dark_matter_profiles.Einasto.F90 (file), 697
dark_matter_profiles.F90 (file), 701
dark_matter_profiles.isothermal.F90 (file), 709
dark_matter_profiles.NFW.F90 (file), 703
dark_matter_profiles.structure.concentration.F90 (file), 711
dark_matter_profiles.structure.concentration.NFW.F90 (file), 713
dark_matter_profiles.structure.concentration.Prada2011.F90 (file), 713
dark_matter_profiles.structure.concentration.Zhao2009.F90 (file), 714
dark_matter_profiles.structure.shape.F90 (file), 715
dark_matter_profiles.structure_tasks.F90 (file), 716
dark_matter_profiles_concentrations (module), 711
dark_matter_profiles_concentrations_gao2008 (module), 712
dark_matter_profiles_concentrations_munozcuartas2011 (module), 712
dark_matter_profiles_concentrations_nfw1996 (module), 713
dark_matter_profiles_concentrations_prada2011 (module), 713
dark_matter_profiles_concentrations_zhao2009 (module), 714
dark_matter_profiles_einasto (module), 697
dark_matter_profiles_einasto_state_retrieve (subroutine), 699
dark_matter_profiles_einasto_state_store (subroutine), 699
dark_matter_profiles_isothermal (module), 709
dark_matter_profiles_nfw (module), 703
dark_matter_profiles_nfw_state_retrieve (subroutine), 705
dark_matter_profiles_nfw_state_store (subroutine), 705
dark_matter_profiles_shapes (module), 715
dark_matter_profiles_shapes_gao2008 (module), 715
dark_matter_shapes_gao2008_initialize (subroutine), 716
dark_matter_shapes_initialize (subroutine), 715
darkmatterprofilecountlinked (function), 932
darkmatterprofilecreatebyinterrupt (subroutine), 932
darkmatterprofilecreatelinked (subroutine), 932
darkmatterprofiledestroylinked (subroutine), 933
darkmatterprofileget (function), 933
darkmatterprofilenullbindingdouble0inout (function), 933
darkmatterprofilenullbindinginteger0in (function), 933
darkmatterprofilenullbindingratedouble0inout (subroutine), 933
darkmatterprofilenullbindingrateinteger0in (subroutine), 933
darkmatterprofilenullbindingsetdouble0inout (subroutine), 933
darkmatterprofilenullbindingsetinteger0in (subroutine), 933
darkmatterprofilescalegrowthrate (function), 934
darkmatterprofilescalegrowthrateisgettable (function), 934
darkmatterprofilescalescalegrowthrateisgettable (function), 934
darkmatterprofilescalescalegrowthratecount (function), 934
darkmatterprofilescalescalegrowthrateget (function), 934
darkmatterprofilescalescalegrowthrategetvalue (function), 934
darkmatterprofilescalescalegrowthratesetgettable (function), 934
darkmatterprofilescalescalegrowthrateset (subroutine), 935
darkmatterprofilescalescaleratex (subroutine), 935
darkmatterprofilescalescalerate (subroutine), 935
CODE INDEX

darkmatterprofilescalescaleset (subroutine), 935
darkmatterprofilescaleshapeshapeset (function), 935
darkmatterprofilescaleshapeshapeget (function), 935
darkmatterprofilescaleshapeshapegrowtherateget (function), 935
darkmatterprofilescaleshapeshapegrowtherateset (subroutine), 935
darkmatterprofilescaleshapeshapegrowtherate (subroutine), 936
darkmatterprofilescaleshapeshapescale (subroutine), 936
darkmatterprofilescaleshapeshapeset (subroutine), 936
darkmatterprofileshe (function), 936
darkmatterprofileshegrowtherate (function), 936
darkmatterprofileshegrowtherateisgettable (function), 936
darkmatterprofilesheisgettable (function), 936
dates_and_times (module), 1320
deadlock_add_node (subroutine), 846
deadlock_tree_output (subroutine), 846
deadlocklist (type), 846
dealloc_array (interface), 1350
dealloc_array_character_1d (subroutine), 1351
dealloc_array_double_precision_1d (subroutine), 1351
dealloc_array_double_precision_2d (subroutine), 1351
dealloc_array_double_precision_3d (subroutine), 1351
dealloc_array_double_precision_4d (subroutine), 1351
dealloc_array_double_precision_5d (subroutine), 1351
dealloc_array_integer_1d (subroutine), 1352
dealloc_array_integer_2d (subroutine), 1352
dealloc_array_integer_kind_int8_1d (subroutine), 1352
dealloc_array_integer_kind_int8_2d (subroutine), 1352
dealloc_array_logical_1d (subroutine), 1352
dealloc_array_real_kind_quad_1d (subroutine), 1352
delete_integer_scalar_ch (subroutine), 1322
delete_integer_scalar_vs (subroutine), 1322
delta (function), 1251
density_einastoscale_free (function), 699
density_nfw_scale_free (function), 705
density_normalization_factor (function), 786
depsln (function), 646
derf (function), 795
derfc0 (function), 795
derfc1 (function), 796
derfi (function), 796
descendent_node_sort_index (function), 842
deserializefromarrayrates (subroutine), 936
deserializefromarrayscales (subroutine), 936
deserializefromarrayvalues (subroutine), 937
destroy_node_indices (subroutine), 842
destroy_vs (subroutine), 813
dgam1 (function), 796
dgamln (function), 796
dgamma (function), 796
dgmln1 (function), 796
diskabundancesgas (function), 937
diskabundancesgasgettable (function), 937
diskabundancesgasrate (subroutine), 937
diskabundancesstellar (function), 937
diskabundancesstellarisgettable (function), 937
diskangularmomentum (function), 937
diskangularmomentumisgettable (function), 937
diskangularmomentumrate (subroutine), 938
diskcountlinked (function), 938
diskcreatebyinterrupt (subroutine), 938
diskcreatelinked (subroutine), 938
diskdestroylinked (subroutine), 938
diskexponentialabundancesgascount (function), 938
diskexponentialabundancesgasget (function), 938
diskexponentialabundancesgasrate (subroutine), 938
diskexponentialabundancesgasrategeneric (subroutine), 938
diskexponentialabundancesgasscale (subroutine), 939
diskexponentialabundancesgasset (subroutine), 939
diskexponentialabundancesstellarcount (function), 939
diskexponentialabundancesstellarget (function), 939
diskexponentialabundancesstellarrate (subroutine), 939
diskexponentialabundancesstellarscale (subroutine), 939
diskexponentialabundancesstellarset (subroutine), 939
diskexponentialangularmomentumcount (function), 939
diskexponentialangularmomentumget (function), 940
diskexponentialangularmomentumrate (subroutine), 940
diskexponentialangularmomentumrategeneric (subroutine), 940
diskexponentialangularmomentumscale (subroutine), 940
diskexponentialangularmomentumset (subroutine), 940
diskexponentialisinitializedget (function), 940
diskexponentialisinitializedset (subroutine), 940
diskexponentialluminositiesstellarcount (function), 940
diskexponentialluminositiesstellarget (function), 940
diskexponentialluminositiesstellarrate (subroutine), 941
diskexponentialluminositiesstellarscale (subroutine), 941
diskexponentialluminositiesstellarset (subroutine), 941
diskexponentialmassgascount (function), 941
diskexponentialmassgasget (function), 941
diskexponentialmassgasrate (subroutine), 941
diskexponentialmassgasrategeneric (subroutine), 941
diskexponentialmassgasscale (subroutine), 941
diskexponentialmassgasset (subroutine), 942
diskexponentialmassstellarcount (function), 942
diskexponentialmassstellarget (function), 942
diskexponentialmassstellarrate (subroutine), 942
diskexponentialmassstellarscale (subroutine), 942
diskexponentialmassstellarset (subroutine), 942
diskexponentialradiusget (function), 942
diskexponentialradiusset (subroutine), 942
diskexponentialstarformationhistorycount (function), 942
diskexponentialstarformationhistoryget (function), 943
diskexponentialstarformationhistoryrate (subroutine), 943
diskexponentialstarformationhistoryscale (subroutine), 943
diskexponentialstarformationhistoryset (subroutine), 943
diskexponentialstarformationrateget (function), 943
diskexponentialstellarpropertieshistorycount (function), 943
diskexponentialstellarpropertieshistoryget (function), 943
diskexponentialstellarpropertieshistoryrate (subroutine), 943
diskexponentialstellarpropertieshistoryscale (subroutine), 944
diskexponentialstellarpropertieshistoryset (subroutine), 944
diskexponentialstellarpropertieshistoryrateget (function), 944
diskexponentialstellarpropertieshistoryset (subroutine), 944
diskget (function), 944
diskhalfmassradius (function), 944
diskhalfmassradiusisgettable (function), 944
diskisinitialized (function), 944
diskisinitializedisgettable (function), 944
diskluminositiesstellar (function), 945
diskluminositiesstellarisgettable (function), 945
diskmassgas (function), 945
diskmassgasisgettable (function), 945
diskmassgasrate (subroutine), 945
diskmassstar (function), 945
diskmassstellarisgettable (function), 945
disknullbindingabundances0inout (function), 945
disknullbindingdouble0inout (function), 946
disknullbindingdouble1inout (function), 946
disknullbindinghistory0inout (function), 946
disknullbindinginteger0in (function), 946
disknullbindinglogical0inout (function), 946
disknullbindingrateabundances0inout (subroutine), 946
disknullbindingratedouble0inout (subroutine), 946
disknullbindingratedouble1inout (subroutine), 946
disknullbindingratehistory0inout (subroutine), 946
disknullbindingrateinteger0in (subroutine), 946
disknullbindingrateinteger0in (subroutine), 946
disknullbindingsetabundances0inout (subroutine), 947
disknullbindingsetdouble0inout (subroutine), 947
disknullbindingsetdouble1inout (subroutine), 947
disknullbindingsethistory0inout (subroutine), 947
disknullbindingsetinteger0in (subroutine), 947
disknullbindingsetlogical0inout (subroutine), 947
disknullbindingsetabundances0inout (subroutine), 947
disknullbindingsetdouble0inout (subroutine), 947
disknullbindingsetdouble1inout (subroutine), 947
disknullbindingsethistory0inout (subroutine), 947
disknullbindingsetinteger0in (subroutine), 947
disknullbindingsetlogical0inout (subroutine), 947
diskradius (function), 947
diskradiusisgettable (function), 947
diskstarformationhistory (function), 947
diskstarformationhistoryisgettable (function), 948
diskstarformationrate (function), 948
diskstarformationrategetfunction (subroutine), 948
diskstarformationrategetisattached (function), 948
diskstarformationrateisgettable (function), 948
diskstellarpropertieshistory (function), 948
diskstellarpropertieshistoryisgettable (function), 948
diskvelocity (function), 948
diskvelocityisgettable (function), 948
diskverysimplemassgascount (function), 949
diskverysimplemassgasset (function), 949
diskverysimplemassgassetgeneric (subroutine), 949
diskverysimplemassgasscale (subroutine), 949
diskverysimplemassgasset (subroutine), 949
diskverysimplemassstellarcount (function), 949
diskverysimplemassstellarget (function), 949
diskverysimplemassstellarrate (subroutine), 950
diskverysimplemassstellaryscale (subroutine), 950
diskverysimplemassstellarset (subroutine), 950
dlnrel (function), 796
do_interpolation (function), 642, 648
double_buffer_dump (subroutine), 757
dpdel (function), 796
dpmpar (function), 646
dpni (subroutine), 796
drcmp (function), 796
drexp (function), 796
drlog (function), 796
dsin1 (function), 797
dump_tree (subroutine), 842
dxparg (function), 646
dynamical_friction_boylankolchin2008 (module), 1164
dynamical_friction_jiang2008 (module), 1164
dynamical_friction_lacey_cole (module), 1165
dynamical_friction_lacey_cole_tormen (module), 1165
dynamical_friction_timescale_multiplier (function), 1168
dynamical_friction_timescale_utilities (module), 1168
dynamical_friction_wetzel_white (module), 1167
early_time_density_scaling (subroutine), 675
early_time_density_scaling_matter_dark_energy (subroutine), 678
early_time_density_scaling_matter_lambda (subroutine), 681
electron_density (function), 643
electron_density_atomic_cie_cloudy (function), 645
electron_density_cie_file (function), 642
electron_density_cie_file_interpolate (function), 642
electron_density_cie_file_logtemperature_interpolate (function), 642
electron_density_density_log_slope (function), 643
electron_density_density_log_slope_atomic_cie_cloudy (function), 645
electron_density_density_log_slope_cie_file (function), 642
electron_density_temperature_log_slope (function), 643
electron_density_temperature_log_slope_atomic_cie_cloudy (function), 646
electron_density_temperature_log_slope_cie_file (function), 642
enclosed_mass_einasto_scale_free (function), 699
enclosed_mass_nfw_scale_free (function), 707
enclosed_mass_root (function), 722
energy_table_make (subroutine), 699
enforce_subhalo_status (subroutine), 842
epoch_of_matter_curvature_equality (function), 675
epoch_of_matter_curvature_equality_matter_dark_energy (function), 678
epoch_of_matter_curvature_equality_matter_lambda (function), 681
epoch_of_matter_dark_energy_equality (function), 675
epoch_of_matter_dark_energy_equality_matter_dark_energy (function), 678
epoch_of_matter_dark_energy_equality_matter_lambda (function), 681
epoch_of_matter_domination (function), 675
epoch_of_matter_domination_matter_dark_energy (function), 678
epoch_of_matter_domination_matter_lambda (function), 682
epsln (function), 646
equivalent_circular_orbit_solver (function), 1180
erfapproximation (function), 1248
error_function (function), 824
error_function_complementary (function), 824
error_functions (module), 824
establish_property_names (subroutine), 757
event_halo_formation (subroutine), 718
events.branch_jump.F90 (file), 717
events.halo_formation.F90 (file), 717
events.node_merger.single_level_hierarchy.F90 (file), 718
events.node_promotion.index_shift.F90 (file), 718
events.subhalo_promotion.F90 (file), 719
events_halo_formation (module), 717
events_node_merger (subroutine), 848
events_node_merger_do_slh (subroutine), 718
events_node_merger_initialize_slh (subroutine), 718
events_node_mergers_slh (module), 718
evolve_to_time (function), 846
evolve_to_time_report (subroutine), 852
evolve_to_time_reports (module), 852
Excursion_Sets.F90 (file), 599
excursion_sets_barrier (function), 1243
excursion_sets_barrier_critical_overdensity (function), 1244
excursion_sets_barrier_effective (function), 1248, 1252
excursion_sets_barrier_gradient (function), 1243
excursion_sets_barrier_gradient_critical_overdensity (function), 1244
excursion_sets_barrier_gradient_linear (function), 1244
excursion_sets_barrier_gradient_quadratic (function), 1245
excursion_sets_barrier_gradient_remap_null (subroutine), 1246
excursion_sets_barrier_gradient_remap_scale (subroutine), 1247
excursion_sets_barrier_gradient_remap_smt (subroutine), 1245
excursion_sets_barrier_initialize (subroutine), 1243
excursion_sets_barrier_linear (function), 1244
excursion_sets_barrier_name (function), 1243
excursion_sets_barrier_quadratic (function), 1245
excursion_sets_barrier_remap_null (subroutine), 1246
excursion_sets_barrier_remap_scale (subroutine), 1247
excursion_sets_barrier_remap_smt (subroutine), 1245
excursion_sets_barriers (module), 1242
excursion_sets_barriers_critical_overdensity (module), 1243
excursion_sets_barriers_critical_overdensity_initialize (subroutine), 1244
excursion_sets_barriers_linear (module), 1244
excursion_sets_barriers_linear_initialize (subroutine), 1244
excursion_sets_barriers_quadratic (module), 1245
excursion_sets_barriers_quadratic_initialize (subroutine), 1245
excursion_sets_barriers_remap_null (module), 1246
excursion_sets_barriers_remap_null_initialize (subroutine), 1246
excursion_sets_barriers_remap_scale (module), 1246
excursion_sets_barriers_remap_scale_initialize (subroutine), 1247
excursion_sets_barriers_remap_smt (module), 1245
excursion_sets_barriers_remap_smt_initialize (subroutine), 1246
excursion_sets_first_crossing_farahii (module), 1248
excursion_sets_first_crossing_farahii_initialize (subroutine), 1248
excursion_sets_first_crossing_farahii_read_file (subroutine), 1248
excursion_sets_first_crossing_farahii_write_file (subroutine), 1249
excursion_sets_first_crossing_linear_barrier (module), 1252
excursion_sets_first_crossing_linear_barrier_initialize (subroutine), 1252
excursion_sets_first_crossing_probability (function), 1247
excursion_sets_first_crossing_probability_farahii (function), 1249
excursion_sets_first_crossing_probability_linear (function), 1253
excursion_sets_first_crossing_probability_zhang_hui (function), 1250
excursion_sets_first_crossing_probability_zhang_hui_high (function), 1252
excursion_sets_first_crossing_rate (function), 1247
excursion_sets_first_crossing_rate_farahii (function), 1249
excursion_sets_first_crossing_rate_linear (function), 1253
excursion_sets_first_crossing_rate_tabulate_farahii (subroutine), 1249
excursion_sets_first_crossing_rate_zhang_hui (function), 1250
excursion_sets_first_crossing_rate_zhang_hui_high (function), 1252
excursion_sets_first_crossing_zhang_hui (module), 1250
excursion_sets_first_crossing_zhang_hui_high (module), 1251
excursion_sets_first_crossing_zhang_hui_high_initialize (subroutine), 1252
excursion_sets_first_crossing_zhang_hui_initialize (subroutine), 1250
excursion_sets_first_crossing_zhang_hui_utilities (module), 1250
excursion_sets_first_crossings (module), 1247
excursion_sets_first_crossings_initialize (subroutine), 1247
excursion_sets_maximum_sigma_test (subroutine), 829
excursion_sets_non_crossing_rate (function), 1248
excursion_sets_non_crossing_rate_farahii (function), 1249
excursion_sets_non_crossing_rate_linear (function), 1253
excursion_sets_non_crossing_rate_zhang_hui (function), 1250
excursion_sets_non_crossing_rate_zhang_hui_high (function), 1252
exists_integer scalar ch (function), 1322
exists_integer scalar vs (function), 1322
expansion factor (function), 675
expansion factor at formation (function), 687
expansion factor c (function), 675
expansion factor change (function), 678
expansion factor from redshift (function), 675
expansion factor is valid (function), 675
expansion factor is valid matter dark energy (function), 678
expansion factor is valid matter lambda (function), 682
expansion factor matter dark energy (function), 678
expansion factor matter lambda (function), 682
expansion rate (function), 675
expansion rate matter dark energy (function), 679
expansion rate matter lambda (function), 682
expansion rate perturbation (function), 1272
exparg (function), 646
exponential integrals (module), 824
extract (interface), 813
extract ch (function), 813
extract vs (function), 813
factorial (function), 825
factorials (module), 825
fftw3 (module), 871
fftw wavenumber (function), 871
fgabnd (function), 601
file exists (interface), 1321
file exists char (function), 1321
file exists varstr (function), 1321
file utilities (module), 1320
filter extent (function), 797
filter get index (function), 797
filter lightcone get coordinates (function), 762
filter luminosity integrand (function), 1225
filter luminosity integrand ab (function), 1226
filter response (function), 797
filter response load (subroutine), 797
filter type (type), 798
fodeiv2 (module), 859
fodeiv2 control (type), 859
fodeiv2 control free (subroutine), 859
fodeiv2 control hadjust (function), 859
fodeiv2 control init (function), 859
fodeiv2 control name (function), 860
fodeiv2 control scaled new (function), 860
fodeiv2 control standard new (function), 860
fodeiv2 control status (function), 860
fodeiv2 control y new (function), 860
fodeiv2 control yp new (function), 860
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Type/Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>fodeiv2_driver</td>
<td>(type), 860</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_driver_alloc_scaled_new</td>
<td>(function), 860</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_driver_alloc_standard_new</td>
<td>(function), 860</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_driver_alloc_y_new</td>
<td>(function), 860</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_driver_alloc_yp_new</td>
<td>(function), 860</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_driver_apply</td>
<td>(function), 860</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_driver_free</td>
<td>(subroutine), 861</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_driver_reset</td>
<td>(function), 861</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_driver_set_hmax</td>
<td>(function), 861</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_driver_set_hmin</td>
<td>(function), 861</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_driver_set_nmax</td>
<td>(function), 861</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_driver_status</td>
<td>(function), 861</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_evolve</td>
<td>(type), 861</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_evolve_alloc</td>
<td>(function), 861</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_evolve_apply</td>
<td>(function), 861</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_evolve_free</td>
<td>(subroutine), 861</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_evolve_reset</td>
<td>(function), 861</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_evolve_status</td>
<td>(function), 861</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_step</td>
<td>(type), 861</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_step_alloc</td>
<td>(function), 862</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_step_apply</td>
<td>(function), 862</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_step_free</td>
<td>(subroutine), 862</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_step_name</td>
<td>(function), 862</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_step_order</td>
<td>(function), 862</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_step_reset</td>
<td>(function), 862</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_step_status</td>
<td>(function), 862</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_step_type</td>
<td>(type), 862</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_system</td>
<td>(type), 862</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_system_free</td>
<td>(subroutine), 862</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_system_init</td>
<td>(function), 862</td>
<td></td>
</tr>
<tr>
<td>fodeiv2_system_status</td>
<td>(function), 862</td>
<td></td>
</tr>
<tr>
<td>formationtimecountlinked</td>
<td>(function), 950</td>
<td></td>
</tr>
<tr>
<td>formationtimecreatebyinterrupt</td>
<td>(subroutine), 950</td>
<td></td>
</tr>
<tr>
<td>formationtimecreatelinked</td>
<td>(subroutine), 950</td>
<td></td>
</tr>
<tr>
<td>formationtimedestroylinked</td>
<td>(subroutine), 950</td>
<td></td>
</tr>
<tr>
<td>formationtimeformationtime</td>
<td>(function), 950</td>
<td></td>
</tr>
<tr>
<td>formationtimeisgettable</td>
<td>(function), 950</td>
<td></td>
</tr>
<tr>
<td>formationtimeget</td>
<td>(function), 951</td>
<td></td>
</tr>
<tr>
<td>formationtimenullbindingdouble0inout</td>
<td>(function), 951</td>
<td></td>
</tr>
<tr>
<td>formationtimenullbindinginteger0in</td>
<td>(function), 951</td>
<td></td>
</tr>
<tr>
<td>formationtimenullbindingrateddouble0inout</td>
<td>(subroutine), 951</td>
<td></td>
</tr>
<tr>
<td>formationtimenullbindingrateinteger0in</td>
<td>(subroutine), 951</td>
<td></td>
</tr>
<tr>
<td>formationtimenullbindingsetdouble0inout</td>
<td>(subroutine), 951</td>
<td></td>
</tr>
<tr>
<td>formationtimenullbindingsetinteger0in</td>
<td>(subroutine), 951</td>
<td></td>
</tr>
<tr>
<td>formatted_date_and_time</td>
<td>(function), 1320</td>
<td></td>
</tr>
<tr>
<td>fourier_profile_integrand_einasto</td>
<td>(function), 699</td>
<td></td>
</tr>
<tr>
<td>fourier_profile_table_make</td>
<td>(subroutine), 699</td>
<td></td>
</tr>
<tr>
<td>freefall_radii</td>
<td>(module), 663</td>
<td></td>
</tr>
<tr>
<td>freefall_radii_dark_matter_halo</td>
<td>(module), 664</td>
<td></td>
</tr>
</tbody>
</table>
freefall_radius (function), 663
freefall_radius_dark_matter_halo (function), 664
freefall_radius_dark_matter_halo_initialize (subroutine), 664
freefall_radius_growth_rate (function), 663
freefall_radius_growth_rate_dark_matter_halo (function), 664
freefall_radius_growth_rate_dark_matter_halo_initialize (subroutine), 664
freefall_radius_growth_rate_dark_matter_halo (function), 664
freefall_time_available_halo_formation (function), 665
freefall_time_available_halo_formation_initialize (subroutine), 665
freefall_time_available_increase_rate_halo_formation (function), 665
freefall_time_scale_free (function), 700, 707
freefall_time_scale_free_integrand (function), 707
freefall_time_scale_free_integrand_einasto (function), 700
freefall_times_available_halo_formation (module), 665
tfhmrinverse (function), 784
g_1 (function), 1251
g_2 (function), 1251
g_2_integrand_zhang_hui (function), 1251
g_2_integrated (function), 1251
galactic_dynamics.bar_instability.ELN.F90 (file), 719
galactic_dynamics.bar_instability.F90 (file), 719
galactic_dynamics.bar_instability.null.F90 (file), 720
galactic_dynamics.bar_instabilities (module), 720
galactic_dynamics.bar_instabilities_eln (module), 719
galactic_dynamics.bar_instabilities_eln_initialize (subroutine), 719
galactic_dynamics.bar_instabilities_null (module), 720
galactic_dynamics.bar_instabilities_null_initialize (subroutine), 720
galactic_dynamics.bar_instability_initialize (subroutine), 720
galactic_structure.density.F90 (file), 720
galactic_structure.enclosed_mass.F90 (file), 721
galactic_structure.options.F90 (file), 722
galactic_structure.potential.F90 (file), 724
galactic_structure.radius_solver.adiabatic.F90 (file), 725
galactic_structure.radius_solver.F90 (file), 725
galactic_structure.radius_solver.fixed.F90 (file), 726
galactic_structure.radius_solver.initial_radii.adiabatic.F90 (file), 728
galactic_structure.radius_solver.initial_radii.F90 (file), 727
galactic_structure.radius_solver.initial_radii.static.F90 (file), 729
galactic_structure.radius_solver.linear.F90 (file), 730
galactic_structure.radius_solver.procedures.F90 (file), 731
galactic_structure.radius_solver.simple.F90 (file), 731
galactic_structure.rotation_curve.F90 (file), 732
galactic_structure.rotation_curve.gradient.F90 (file), 732
galactic_structure.surface_density.F90 (file), 733
galactic_structure.velocity_dispersions.F90 (file), 733
galactic_structure_component_type_decode (function), 724
galactic_structure_densities (module), 721
galactic_structure_density (function), 721
galactic_structure_enclosed_mass (function), 722
galactic_structure_enclosed_mass_defaults (subroutine), 722
galactic_structure_enclosed_masses (module), 721
galactic_structure_initial_radii (module), 727
galactic_structure_initial_radii_adiabatic (module), 728
galactic_structure_initial_radii_adiabatic_initialize (subroutine), 728
galactic_structure_initial_radii_static (module), 729
galactic_structure_initial_radii_static_initialize (subroutine), 730
galactic_structure_mass_type_decode (function), 724
galactic_structure_options (module), 723
galactic_structure_potential (function), 725
galactic_structure_potentials (module), 724
galactic_structure_radii (module), 725
galactic_structure_radii_adiabatic (module), 726
galactic_structure_radii_adiabatic_initialize (subroutine), 726
galactic_structure_radii_fixed (module), 726
galactic_structure_radii_fixed_initialize (subroutine), 726
galactic_structure_radii_initial_adiabatic_compute_factors (subroutine), 729
galactic_structure_radii_linear (module), 730
galactic_structure_radii_linear_initialize (subroutine), 730
galactic_structure_radii_simple (module), 731
galactic_structure_radii_simple_initialize (subroutine), 731
galactic_structure_radii_solve (subroutine), 725
galactic_structure_radii_solve_adiabatic (subroutine), 726
galactic_structure_radii_solve_fixed (subroutine), 727
galactic_structure_radii_solve_linear (subroutine), 730
galactic_structure_radii_solve_simple (subroutine), 731
galactic_structure_radius_enclosing_mass (function), 722
galactic_structure_radius_initial (function), 727
galactic_structure_radius_initial_adiabatic (function), 729
galactic_structure_radius_initial_adiabatic_solver (function), 729
galactic_structure_radius_initial_derivative (function), 727
galactic_structure_radius_initial_derivative_adiabatic (function), 729
galactic_structure_radius_initial_derivative_adiabatic_solver (function), 729
galactic_structure_radius_initial_derivative_static (function), 730
galactic_structure_radius_initial_initialize (subroutine), 727
galactic_structure_radius_initial_static (function), 730
galactic_structure_radius_solver_procedures (module), 731
galactic_structure_rotation_curve (function), 732
galactic_structure_rotation_curve_gradient (function), 733
galactic_structure_rotation_curve_gradients (module), 732
galactic_structure_rotation_curves (module), 732
galactic_structure_surface_densities (module), 733
galactic_structure_surface_density (function), 733
galactic_structure_velocity_dispersion (function), 734
galactic_structure_velocity_dispersions (module), 734
galacticus (program), 599
galacticus.banner.F90 (file), 734
galacticus.calculations_reset.F90 (file), 734
galacticus.display.F90 (file), 735
galacticus.error.F90 (file), 739
galacticus_display_message (interface), 739
galacticus_display_message_char (subroutine), 739
galacticus_display_message_varstr (subroutine), 739
galacticus_display_unindent (subroutine), 739
galacticus_error (module), 740
galacticus_error_handler_register (subroutine), 751
galacticus_error_report (interface), 751
galacticus_error_report_char (subroutine), 751
galacticus_error_report_varstr (subroutine), 751
galacticus_extra_output_halo_fourier_profile (subroutine), 765
galacticus_growth_factor_output (subroutine), 765
galacticus_gsl_error_handler (subroutine), 751
galacticus_hdf5 (module), 755
galacticus_input_path (function), 752
galacticus_input_paths (module), 752
galacticus_linear_power_spectrum_output (subroutine), 765
galacticus_merger_tree_output (subroutine), 758
galacticus_merger_tree_output_filter (function), 761
galacticus_merger_tree_output_filter_initialize (subroutine), 761
galacticus_merger_tree_output_filter_lightcone (subroutine), 762
galacticus_merger_tree_output_filter_lightcone_initialize (subroutine), 762
galacticus_merger_tree_output_filter_lightcones (module), 761
galacticus_merger_tree_output_filter_luminosities (module), 763
galacticus_merger_tree_output_filter_luminosity (subroutine), 763
galacticus_merger_tree_output_filter_luminosity_initialize (subroutine), 763
galacticus_merger_tree_output_filter_stellar_mass (subroutine), 763
galacticus_merger_tree_output_filter_stellar_mass_initialize (subroutine), 764
galacticus_merger_tree_output_filter_stellar_masses (module), 763
galacticus_merger_tree_output_filters (module), 761
galacticus_merger_tree_output_finalize (subroutine), 758
galacticus_merger_tree_output_make_group (subroutine), 758
galacticus_meta_compute_times (module), 753
galacticus_meta_compute_times_file (module), 753
galacticus_meta_evolver_profile (subroutine), 754
galacticus_meta_evolver_profiler (module), 754
galacticus_meta_evolver_profiler_output (subroutine), 754
galacticus_meta_tree_timing (module), 754
galacticus_next_output_time (function), 777
galacticus_nodes (module), 914
galacticus_nodes_finalize (subroutine), 951
galacticus_nodes_initialize (subroutine), 951
galacticus_nodes_unique_id_set (subroutine), 952
galacticus_output_close_file (subroutine), 756
galacticus_output_halo_model (subroutine), 765
galacticus_output_halo_model_initialize (subroutine), 766
galacticus_output_halo_model_names (subroutine), 766
galacticus_output_halo_model_property_count (subroutine), 766
galacticus_output_halo_models (module), 765
galacticus_output_merger_tree (module), 756
galacticus_output_most_massive_progenitor (subroutine), 768
galacticus_output_most_massive_progenitor_initialize (subroutine), 768
galacticus_output_most_massive_progenitor_names (subroutine), 769
galacticus_output_most_massive_progenitor_property_count (subroutine), 769
galacticus_output_most_massive_progenitors (module), 768
galacticus_output_open (module), 755
galacticus_output_open_file (subroutine), 756
galacticus_output_star_formation_histories (module), 771
galacticus_output_star_formation_histories_initialize (subroutine), 771
galacticus_output_time (function), 777
galacticus_output_time_count (function), 777
galacticus_output_time_index (function), 777
galacticus_output_times (module), 777
galacticus_output_tree_density_contrast (subroutine), 759
galacticus_output_tree_density_contrast_initialize (subroutine), 759
galacticus_output_tree_density_contrast_names (subroutine), 760
galacticus_output_tree_density_contrast_property_count (subroutine), 760
galacticus_output_tree_descendants (subroutine), 760
galacticus_output_tree_descendants_initialize (subroutine), 760
galacticus_output_tree_descendants_names (subroutine), 761
galacticus_output_tree_descendants_property_count (subroutine), 761
galacticus_output_tree_half_light (subroutine), 764
galacticus_output_tree_half_light_initialize (subroutine), 764
galacticus_output_tree_half_light_names (subroutine), 764
galacticus_output_tree_half_light_properties (module), 764
galacticus_output_tree_half_light_property_count (subroutine), 764
galacticus_output_tree_lightcone (subroutine), 762
galacticus_output_tree_lightcone_names (subroutine), 762
galacticus_output_tree_lightcone_property_count (subroutine), 762
galacticus_output_tree_links (subroutine), 766
galacticus_output_tree_links_names (subroutine), 766
galacticus_output_tree_links_property_count (subroutine), 766
galacticus_output_tree_main_branch (subroutine), 767
galacticus_output_tree_main_branch_initialize (subroutine), 767
galacticus_output_tree_main_branch_names (subroutine), 767
galacticus_output_tree_main_branch_property_count (subroutine), 767
galacticus_output_tree_mass_profile (subroutine), 768
galacticus_output_tree_mass_profile_initialize (subroutine), 768
galacticus_output_tree_mass_profile_names (subroutine), 768
galacticus_output_tree_mass_profile_property_count (subroutine), 768
galacticus_output_tree_mass_profiles (module), 767
galacticus_output_tree_rotation_curve (subroutine), 769
galacticus_output_tree_rotation_curve_initialize (subroutine), 769
galacticus_output_tree_rotation_curve_names (subroutine), 769
galacticus_output_tree_rotation_curve_property_count (subroutine), 770
galacticus_output_tree_satellite_pericenter (subroutine), 770
galacticus_output_tree_satellite_pericenter_initialize (subroutine), 770
galacticus_output_tree_satellite_pericenter_names (subroutine), 770
galacticus_output_tree_satellite_pericenter_property_count (subroutine), 770
galacticus_output_tree_velocity_dispersion (subroutine), 774
galacticus_output_tree_velocity_dispersion_initialize (subroutine), 774
galacticus_output_tree_velocity_dispersion_names (subroutine), 775
galacticus_output_tree_velocity_dispersion_property_count (subroutine), 775
galacticus_output_tree_virial (subroutine), 776
galacticus_output_tree_virial_initialize (subroutine), 776
galacticus_output_tree_virial_names (subroutine), 776
galacticus_output_tree_virial_property_count (subroutine), 776
galacticus_output_trees_density_contrasts (module), 759
galacticus_output_trees_descendents (module), 760
galacticus_output_trees_line_of_sight_velocity_dispersion (function), 775
galacticus_output_trees_links (module), 766
galacticus_output_trees_main_branch (module), 767
galacticus_output_trees_rotation_curve (module), 769
galacticus_output_trees_satellite_pericenter (module), 770
galacticus_output_trees_velocity_dispersion (module), 774
galacticus_output_trees_velocity Dispersion_densInteg (function), 775
galacticus_output_trees_virial (module), 776
galacticus_output_trees_vlcty Dispersion_vlcty_dnsty_integrand (function), 775
galacticus_output_trees_vlcty Dispersion_vlcty_dnsty_srnc_integrand (function), 775
galacticus_output_trees_vlcty Dispersion_vlcty_dnsty_srfc_integrand (function), 775
galacticus_previous_output_time (function), 777
galacticus_signal_handler_sigfpe (subroutine), 751
galacticus_signal_handler_sigint (subroutine), 751
galacticus_signal_handler_sigsegv (subroutine), 752
galacticus_state (module), 778
galacticus_state_retrieve (subroutine), 779
galacticus_state_snapshot (subroutine), 779
galacticus_task_do (subroutine), 780
galacticus_task_end (function), 780
galacticus_task_evolve_tree (function), 781
galacticus_task_start (function), 780
galacticus_tasks (module), 780
galacticus_tasks_basic (module), 780
galacticus_time_per_tree (module), 781
galacticus_time_per_tree (function), 753
galacticus_time_per_tree_file (function), 753
galacticus_time_per_tree_file_initialize (subroutine), 753
galacticus_time_per_tree_initialize (subroutine), 753

galacticus_time_per_tree_velocity Dispersion_vlcty_dnsty_integrand (function), 739
galacticus_time_per_tree_velocity Dispersion_vlcty_dnsty_srnc_integrand (function), 739

gamminv (subroutine), 797
gamma_function (function), 825
gamma_function_incomplete (function), 826
gamma_function_incomplete_complementary (function), 826
gamma_function_logarithmic (function), 826
gamma_functions (module), 825
gaussian_distribution (function), 824
gaussian_random (module), 877
gaussian_random_free (subroutine), 877
gaussian_random_get (function), 877
generalized_press_schechter_branch_mass (function), 829
generalized_press_schechter_branch_mass_root (function), 829
generalized_press_schechter_branching (module), 829
generalized_press_schechter_branching_initialize (subroutine), 830
generalized_press_schechter_branching_maximum_step (function), 830
generalized_press_schechter_branching_probability (function), 830
generalized_press_schechter_subresolution_fraction (function), 830
genericnullbindingabundances0inout (function), 952
genericnullbindingdouble0inout (function), 952
genericnullbindingrateabundances0inout (subroutine), 952
genericnullbindingratedouble0inout (subroutine), 952
genericnullbindingsetabundances0inout (subroutine), 952
genericnullbindingsetdouble0inout (subroutine), 952
generalized_press_schechter_setdouble0inout (subroutine), 952
geometry.coordinate_systems.F90 (file), 781
get (interface), 813
get_ (subroutine), 813
get_argument (interface), 1319
get_argument_character (subroutine), 1319
get_argument_double (subroutine), 1319
get_argument_integer (subroutine), 1319
get_argument_logical (subroutine), 1319
get_argument_real (subroutine), 1319
get_argument_varying_string (subroutine), 1320
get_chemical_masses (subroutine), 606
get_input_parameter (interface), 1331
get_input_parameter_array_size (function), 1331
get_input_parameter_char (subroutine), 1331
get_input_parameter_char_array (subroutine), 1332
get_input_parameter_double (subroutine), 1332
get_input_parameter_double_array (subroutine), 1332
get_input_parameter_double_c (subroutine), 1332
get_input_parameter_integer (subroutine), 1332
get_input_parameter_integer_array (subroutine), 1333
get_input_parameter_integer_c (subroutine), 1334
get_input_parameter_integer_long (subroutine), 1334
get_input_parameter_integer_long_array (subroutine), 1334
get_input_parameter_logical (subroutine), 1334
get_input_parameter_logical_array (subroutine), 1334
get_input_parameter_varstring (subroutine), 1335
get_input_parameter_varstring_array (subroutine), 1336
get_interpolation (subroutine), 642, 648
get_new_assert_result (function), 1362
get_path (subroutine), 752

1452
get_set_ch (subroutine), 813
get_set_vs (subroutine), 813
get_temporary_string (subroutine), 1320
get_tree (subroutine), 781
get_unit (subroutine), 813
get_unit_set_ch (subroutine), 814
get_unit_set_vs (subroutine), 814
gratio (subroutine), 797
growing_core_virial_density_function (function), 788
growth_factor_simple_initialize (subroutine), 1261
growthrateodes (function), 688
growthtableodes (function), 1261
h_0 (function), 685
h_0_invgyr (function), 685
h_electron_to_hminus_photon_rate_coefficient (function), 639
h_hminus_to_h2_electron_rate_coefficient (function), 639
half_mass_radius_root (function), 842
half_mass_radius_root_cole2000 (function), 1174
halo_baryonic_accreted_abundances (subroutine), 604
halo_baryonic_accreted_abundances_null_get (subroutine), 605
halo_baryonic_accreted_abundances_simple_get (subroutine), 607
halo_baryonic_accreted_chemicals (subroutine), 604
halo_baryonic_accreted_chemicals_null_get (subroutine), 605
halo_baryonic_accreted_chemicals_simple_get (subroutine), 607
halo_baryonic_accreted_mass (function), 604
halo_baryonic_accreted_mass_null_get (function), 605
halo_baryonic_accreted_mass_simple_get (function), 607
halo_baryonic_accretion_rate (function), 604
halo_baryonic_accretion_rate_abundances (subroutine), 604
halo_baryonic_accretion_rate_abundances_null_get (subroutine), 605
halo_baryonic_accretion_rate_abundances_simple_get (subroutine), 607
halo_baryonic_accretion_rate_chemicals (subroutine), 604
halo_baryonic_accretion_rate_chemicals_null_get (subroutine), 605
halo_baryonic_accretion_rate_chemicals_simple_get (subroutine), 607
halo_baryonic_accretion_rate_null_get (function), 606
halo_baryonic_accretion_rate_simple_get (function), 607
halo_baryonic_failed_accreted_mass (function), 604
halo_baryonic_failed_accreted_mass_null_get (function), 606
halo_baryonic_failed_accreted_mass_simple_get (function), 607
halo_baryonic_failed_accretion_rate (function), 604
halo_baryonic_failed_accretion_rate_null_get (function), 606
halo_baryonic_failed_accretion_rate_simple_get (function), 608
halo_mass_fraction_integrand (function), 1256
halo_mass_fraction_integrated (function), 1256
halo_mass_function (module), 1256
halo_mass_function_close_file (subroutine), 782
halo_mass_function_compute (subroutine), 782
halo_mass_function_differential (function), 1256
halo_mass_function_differential_press_schechter (function), 1257
halo_mass_function_differential_tinker2008 (function), 1258
halo_mass_function_initialize (subroutine), 1256
halo_mass_function_integrand (function), 1256
halo_mass_function_integrated (function), 1257
halo_mass_function_open_file (subroutine), 783
halo_mass_function_output (subroutine), 783
halo_mass_function_press_schechter (module), 1257
halo_mass_function_press_schechter_initialize (subroutine), 1257
halo_mass_function_sheth_tormen (module), 1257
halo_mass_function_sheth_tormen_differential (function), 1257
halo_mass_function_sheth_tormen_initialize (subroutine), 1258
halo_mass_function_tasks (module), 782
halo_mass_function_tinker2008 (module), 1258
halo_mass_function_tinker2008_initialize (subroutine), 1258
halo_mass_functions (program), 600
HalloMassFunctions.F90 (file), 600
halo_mass_functions.tasks.F90 (file), 782
halo_model.conditional_stellar_mass_function.Behroozi2010.F90 (file), 783
halo_model.conditional_stellar_mass_function.F90 (file), 784
halo_spin_distribution_bett2007 (function), 694
halo_spin_distribution_bett2007_initialize (subroutine), 694
halo_spin_distribution_bett2007_snapshot (subroutine), 694
halo_spin_distribution_bett2007_state_retrieve (subroutine), 694
halo_spin_distribution_bett2007_state_store (subroutine), 695
halo_spin_distribution_delta_function (function), 696
halo_spin_distribution_delta_function_initialize (subroutine), 696
halo_spin_distribution_lognormal (function), 696
halo_spin_distribution_lognormal_initialize (subroutine), 696
halo_spin_distribution_lognormal_snapshot (subroutine), 696
halo_spin_distribution_lognormal_state_retrieve (subroutine), 696
halo_spin_distribution_lognormal_state_store (subroutine), 697
halo_spin_distribution_sample (function), 695
halo_spin_distributions (module), 695
halo_spin_distributions_bett2007 (module), 694
halo_spin_distributions_delta_function (module), 695
halo_spin_distributions_lognormal (module), 696
halo_spin_sample_get_template (function), 695
halo_spin_sample_get_template (interface), 695
halo_virial_density_contrast (function), 1280
halo_virial_density_contrast_rate_of_change (function), 1280
hash_md5 (function), 1323
hash_perfect_create (subroutine), 1324
hash_perfect_destroy (subroutine), 1324
hash_perfect_index (function), 1324
hash_perfect_is_present (function), 1324
hash_perfect_size (function), 1324
hash_perfect_value (function), 1324
hashes (module), 1322
hashes_cryptographic (module), 1323
hashes_perfect (module), 1324
hashperfect (type), 1325
hdf5object (type), 1304
histories (module), 898
histories_state_retieve (subroutine), 899
histories_state_store (subroutine), 899
history (type), 899
history_add (function), 899
history_builder (subroutine), 900
history_clone (subroutine), 900
history_combine (subroutine), 900
history_create (subroutine), 900
history_deserialize (subroutine), 900
history_destroy (subroutine), 900
history_divide (function), 900
history_dump (subroutine), 901
history_dump_raw (subroutine), 901
history_exists (function), 901
history_extend (subroutine), 901
history_increment (subroutine), 901
history_is_zero (function), 901
history_read_raw (subroutine), 901
history_reset (subroutine), 901
history_serialize (subroutine), 902
history_serialize_count (function), 902
history_set_times (subroutine), 902
history_set_to_unity (subroutine), 902
history_subtract (function), 902
history_timesteps (subroutine), 902
history_trim (subroutine), 902
hminus_electron_to_h_2electron_rate_coefficient (function), 639
hminus_hplus_to_2h_rate_coefficient (function), 639
hot_halo.density_profile.cored_isothermal.core_radius.F90 (file), 787
hot_halo.density_profile.cored_isothermal.core_radius.growing_core.F90 (file), 788
hot_halo.density_profile.cored_isothermal.core_radius.virial_radius_fraction.F90 (file), 789
hot_halo.density_profile.cored_isothermal.F90 (file), 786
hot_halo.density_profile.null.F90 (file), 790
hot_halo.ram_pressure_force.F90 (file), 791
hot_halo.ram_pressure_force.Font2008.F90 (file), 791
hot_halo.ram_pressure_force.null.F90 (file), 792
hot_halo.ram_pressure_stripping.F90 (file), 792
hot_halo.ram_pressure_stripping.Font2008.F90 (file), 793
hot_halo.ram_pressure_stripping.virial_radius.F90 (file), 793
hot_halo.temperature_profile.F90 (file), 794
hot_halo.temperature_profile.virial.F90 (file), 795
hot_halo_density (function), 785
hot_halo_density_cored_isothermal (subroutine), 787
hot_halo_density_cored_isothermal_core_radii (module), 788
hot_halo_ram_pressure_stripping_virial_radius (function), 794
hot_halo_temperature (function), 794
hot_halo_temperature_get_template (function), 794
hot_halo_temperature_get_template (interface), 794
hot_halo_temperature_initialize (subroutine), 794
hot_halo_temperature_logarithmic_slope (function), 794
hot_halo_temperature_logarithmic_slope_virial (function), 794
hot_halo_temperature_logarithmic_slope_virial_get (function), 795
hot_halo_temperature_profile (module), 794
hot_halo_temperature_profile_virial (module), 795
hot_halo_temperature_virial (subroutine), 795
hot_halo_temperature_virial_get (function), 795
hot_mode_fraction (function), 1122
hothaloabundances (function), 952
hothaloabundancesisgettable (function), 952
hothaloangularmomentum (function), 952
hothaloangularmomentumisgettable (function), 953
hothalochemicals (function), 953
hothalochemicalsisgettable (function), 953
hothalocountlinked (function), 953
hothallocreatebyinterrupt (subroutine), 953
hothallocreatelinked (subroutine), 953
hothalodestroylinked (subroutine), 953
hothaloget (function), 953
hothalohothalocoolingabundances (function), 954
hothalohothalocoolingabundancesisgettable (function), 954
hothalohothalocoolingabundancesratefunction (subroutine), 954
hothalohothalocoolingabundancesrateisattached (function), 954
hothalohothalocoolingangularmomentum (function), 954
hothalohothalocoolingangularmomentumisgettable (function), 954
hothalohothalocoolingangularmomentumratefunction (subroutine), 955
hothalohothalocoolingangularmomentumrateisattached (function), 955
hothalohothalocoolingmass (function), 955
hothalohothalocoolingmassisgettable (function), 955
hothalohothalocoolingmassratefunction (subroutine), 955
hothalohothalocoolingmassrateisattached (function), 955
hothalomass (function), 955
hothalomassisgettable (function), 955
hothalomasssink (function), 955
hothalomasssinkisgettable (function), 956
hothalonullbindingabundances0inout (function), 956
hothalonullbindingchemicalabundances0inout (function), 956
hothalonullbindingdouble0inout (function), 956
hothalonullbindinginteger0in (function), 956
hothalonullbindingrateabundances0inout (subroutine), 956
hothalonullbindingratechemicalabundances0inout (subroutine), 956
hothermalnullbindingratedouble0inout (subroutine), 956
hothermalnullbindingrateinteger0in (subroutine), 956
hothermalnullbindingsetabundances0inout (subroutine), 957
hothermalnullbindingsetchemicalabundances0inout (subroutine), 957
hothermalnullbindingsettodouble0inout (subroutine), 957
hothermalnullbindingsetinteger0in (subroutine), 957
hothaloouterradius (function), 957
hothaloouterradiusgetfunction (subroutine), 957
hothaloouterradiusgetisattached (function), 957
hothaloouterradiusisgettable (function), 957
hothalooutflowedabundances (function), 957
hothalooutflowedabundancesisgettable (function), 957
hothalooutflowedangularmomentum (function), 958
hothalooutflowedangularmomentumisgettable (function), 958
hothalooutflowedmass (function), 958
hothalooutflowedmassisgettable (function), 958
hothalooutflowingabundances (function), 958
hothalooutflowingabundancesisgettable (function), 958
hothalooutflowingabundancesratefunction (subroutine), 958
hothalooutflowingabundancesrateisattached (function), 958
hothalooutflowingangularmomentum (function), 959
hothalooutflowingangularmomentumisgettable (function), 959
hothalooutflowingangularmomentumratefunction (subroutine), 959
hothalooutflowingangularmomentumrateisattached (function), 959
hothalooutflowingmass (function), 959
hothalooutflowingmassisgettable (function), 959
hothalooutflowingmassratefunction (subroutine), 959
hothalooutflowingmassrateisattached (function), 959
hothalostandardabundancescount (function), 959
hothalostandardabundancesget (function), 960
hothalostandardabundancesrate (subroutine), 960
hothalostandardabundancesscale (subroutine), 960
hothalostandardabundancesset (subroutine), 960
hothalostandardangularmomentumcount (function), 960
hothalostandardangularmomentumget (function), 960
hothalostandardangularmomentumrate (subroutine), 960
hothalostandardangularmomentumscale (subroutine), 960
hothalostandardangularmomentumset (subroutine), 960
hothalostandardchemicalscount (function), 961
hothalostandardchemicalsget (function), 961
hothalostandardchemicalsrate (subroutine), 961
hothalostandardchemicalsscale (subroutine), 961
hothalostandardchemicalsset (subroutine), 961
hothalostandardheatsourcerate (subroutine), 961
hothalostandardhthalocoolingabundancesrate (subroutine), 961
hothalostandardhthalocoolingangularmomentumrate (subroutine), 961
hothalostandardhthalocoolingmassrate (subroutine), 961
hothalostandardmasscount (function), 962
hothalostandardmassget (function), 962
hothalostandardmassrate (subroutine), 962
hothalostandardmassscale (subroutine), 962
hothalostandardmassset (subroutine), 962
hothalostandardouterradiuscoun (function), 962
hothalostandardouterradiusget (function), 962
hothalostandardouterradiusgetvalue (function), 962
hothalostandardouterradiussrate (subroutine), 962
hothalostandardouterradiusscale (subroutine), 963
hothalostandardouterradiussset (subroutine), 963
hothalostandardoutflowedabundancescount (function), 963
hothalostandardoutflowedabundancesget (function), 963
hothalostandardoutflowedabundancesgetfunction, 963
hothalostandardoutflowedabundancesset (subroutine), 963
hothalostandardoutflowedangularmomentumcount (function), 963
hothalostandardoutflowedangularmomentumget (function), 964
hothalostandardoutflowedangularmomentumsrate (subroutine), 964
hothalostandardoutflowedangularmomentumscale (subroutine), 964
hothalostandardoutflowedangularmomentumsset (subroutine), 964
hothalostandardoutflowedmasscount (function), 964
hothalostandardoutflowedmassget (function), 964
hothalostandardoutflowedmassrate (subroutine), 964
hothalostandardoutflowedmassscale (subroutine), 964
hothalostandardoutflowedmassset (subroutine), 964
hothalostandardoutflowingabundancesrate (subroutine), 965
hothalostandardoutflowingangularmomentumsrate (subroutine), 965
hothalostandardoutflowingmassrate (subroutine), 965
hothalostandardoutflowingmassescount (function), 965
hothalostandardoutflowingmassesget (function), 965
hothalostandardoutflowingmassesrate (subroutine), 965
hothalostandardoutflowingmassesset (subroutine), 965
hothalostandardunaccretedmasscount (function), 966
hothalostandardunaccretedmassget (function), 966
hothalostandardunaccretedmassrate (subroutine), 966
hothalostandardunaccretedmassscale (subroutine), 966
hothalostandardunaccretedmassset (subroutine), 966
hothalostrippedabundances (function), 967
hothalostrippedabundancesisgettable (function), 967
hothalostrippedmass (function), 967
hothalostrippedmassisgettable (function), 967
hothalounaccretedmass (function), 967
hothalounaccretedmassisgettable (function), 967
hothaloverysimplehothalocoolingmassrate (subroutine), 967
hothaloverysimplemasscount (function), 967
hothaloverysimplemassget (function), 968
hothaloverysimplemassrate (subroutine), 968
hothaloverysimplemassscale (subroutine), 968
hothaloverysimplemassset (subroutine), 968
hothaloverysimpleoutflowingmassrate (subroutine), 968
hubble_parameter (function), 676
hubble_parameter_matter_dark_energy (function), 679
hubble_parameter_matter_lambda (function), 682
hypergeometric_1f1 (function), 826
hypergeometric_2f1 (function), 827
hypergeometric_functions (module), 826
iachar (interface), 814
iachar_ (function), 814
ichar (interface), 814
ichar_ (function), 814
idbvip (subroutine), 620
idcldp (subroutine), 621
ideal_gas_jeans_length (function), 1303
ideal_gas_sound_speed (function), 1303
ideal_gases_thermodynamics (module), 1303
idgrid (subroutine), 621
idictn (subroutine), 621
idpdrv (subroutine), 621
idptip (subroutine), 621
idsfft (subroutine), 621
idtang (subroutine), 621
idxchg (function), 621
igm_state_electron_scattering_integrand (function), 798
igm_state_electron_scattering_tabulate (subroutine), 798
imf_descriptor (function), 1184
imf_energy_input_rate_noninstantaneous (function), 1184
imf_index_lookup (function), 1196
imf_maximum_mass (function), 1184
imf_metal_yield_rate_noninstantaneous (function), 1185
imf_minimum_mass (function), 1185
imf_name (function), 1185
imf_phi (function), 1185
imf_recycled_fraction_instantaneous (function), 1185
imf_recycling_rate_noninstantaneous (function), 1185
imf_select (function), 1187
imf_select_disk_spheroid (function), 1195
imf_select_disk_spheroid_initialize (subroutine), 1195
imf_select_fixed (function), 1196
imf_select_fixed_initialize (subroutine), 1196
imf_tabulate (subroutine), 1187
imf_yield_instantaneous (function), 1187
inc_gam.F90 (file), 795
incomplete_gamma (module), 795

1460
index (interface), 814
index_ch_vs (function), 814
index_vs_ch (function), 814
index_vs_vs (function), 814
indicesbranchtip (function), 968
indicesbranchtipisgettable (function), 968
indicescountlinked (function), 968
indicescreatebyinterrupt (subroutine), 968
indicescreatelinked (subroutine), 969
indicesdestroylinked (subroutine), 969
indicesget (function), 969
indicesnullbindinginteger0in (function), 969
indicesnullbindinglonginteger0inout (function), 969
indicesnullbindingrateinteger0in (subroutine), 969
indicesnullbindingratelonginteger0inout (subroutine), 969
indicesnullbindingsetinteger0in (subroutine), 969
indicesnullbindingratelonginteger0inout (subroutine), 969
indicesstandardbranchtipget (function), 970
indicesstandardbranchtipset (subroutine), 970
infall_radii_cooling_freefall (module), 666
infall_radii_cooling_radius (module), 667
infall_radius (function), 666
infall_radius_cooling_freefall (function), 666
infall_radius_cooling_freefall_initialize (subroutine), 667
infall_radius_cooling_radius (function), 667
infall_radius_cooling_radius_initialize (subroutine), 667
infall_radius_cooling_radius_growth_rate (function), 666
infall_radius_growth_rate_cooling_freefall (function), 667
infall_radius_growth_rate_cooling_radius (function), 667
infall_radius_initialize (subroutine), 666
initialize_cosmological_mass_variance (subroutine), 1262
initialize_display (subroutine), 739
initialize_integer_scalar (subroutine), 1322
input_parameter_is_present (function), 1336
input_parameters (module), 1325
input_parameters_file_close (subroutine), 1336
input_parameters_file_open (subroutine), 1336
insert (interface), 814
insert_ch_ch (function), 814
insert_ch_vs (function), 814
insert_vs_ch (function), 815
insert_vs_vs (function), 815
instruments.filters.F90 (file), 797
instruments_filters (module), 797
integer_buffer_dump (subroutine), 759
integerscalarhash (type), 1322
integrand1 (function), 1293
integrand2 (function), 1293
integrand3 (function), 1293
integrand4 (function), 1293
integrate (function), 872
integrate_done (subroutine), 872
integration_gal_error_handler (subroutine), 872
intergalactic_medium_electron_fraction (function), 798
intergalactic_medium_electron_fraction_file (function), 800
intergalactic_medium_electron_scattering_optical_depth (function), 798
intergalactic_medium_electron_scattering_time (function), 799
intergalactic_medium_file_set_file (subroutine), 800
intergalactic_medium_state (module), 798
intergalactic_medium_state_file (module), 800
intergalactic_medium_state_file_current_file_format_version (function), 800
intergalactic_medium_state_file_initialize (subroutine), 800
intergalactic_medium_state_file_read_data (subroutine), 800
intergalactic_medium_state_initialize (subroutine), 799
intergalactic_medium_state_recfast (module), 799
intergalactic_medium_state_recfast_initialize (subroutine), 799
intergalactic_medium_state_state_retrieve (subroutine), 799
intergalactic_medium_state_state_store (subroutine), 799
intergalactic_medium_temperature (function), 799
intergalactic_medium_temperature_file (function), 801
intergalactic_medium_state.F90 (file), 798
intergalactic_medium_state_file.F90 (file), 800
intergalactic_medium_state.RecFast.F90 (file), 799
interoutputcountlinked (function), 970
interoutputcreatebyinterrupt (subroutine), 970
interoutputcreatelinked (subroutine), 970
interoutputdestroylinked (subroutine), 970
interoutputdiskstarformationrate (function), 970
interoutputdiskstarformationrateisgettable (function), 970
interoutputdiskstarformationraterate (subroutine), 970
interoutputget (function), 971
interoutputnullbindingdouble0inout (function), 971
interoutputnullbindinginteger0in (function), 971
interoutputnullbindingratedouble0inout (subroutine), 971
interoutputnullbindingrateinteger0in (subroutine), 971
interoutputnullbindingsetdouble0inout (subroutine), 971
interoutputnullbindingsetinteger0in (subroutine), 971
interoutputspheroidstarformationraterate (function), 971
interoutputspheroidstarformationrateisgettable (function), 971
interoutputspheroidstarformationraterate (subroutine), 972
interoutputstandarddiskstarformationratecount (function), 972
interoutputstandarddiskstarformationrateget (function), 972
interoutputstandarddiskstarformationraterate (subroutine), 972
interoutputstandarddiskstarformationratescale (subroutine), 972
interoutputstandarddiskstarformationrateset (subroutine), 972
interoutputstandardpseudopassive (function), 972
interoutputstandardspheroidstarformationraterateget (function), 972
interoutputstandardspheroidstarformationraterate (subroutine), 973
interoutputstandardspheroidstarformationratescale (subroutine), 973
interoutputstandardspheroidstarformationrateset (subroutine), 973
interp2dirregularobject (type), 873
interpolate (function), 875
interpolate_2d_irregular (interface), 873
interpolate_2d_irregular_array (function), 873
interpolate_2d_irregular_scalar (function), 873
interpolate_derivative (function), 875
interpolate_done (subroutine), 875
interpolate_in_wavenumber (subroutine), 1259
interpolate_linear_do (function), 875
interpolate_linear_generate_factors (function), 875
interpolate_linear_generate_gradient_factors (function), 875
interpolate_locate (function), 875
inverse_gamma_function_incomplete (function), 826
inverse_gamma_function_incomplete_complementary (function), 826
inversesigmamin (function), 714
io_hdf5 (module), 1304
io_hdf5_assert_attribute_type (subroutine), 1304
io_hdf5_assert_dataset_type (subroutine), 1304
io_hdf5_character_types (function), 1305
io_hdf5_close (subroutine), 1305
io_hdf5_create_reference_scalar_to_1d (subroutine), 1305
io_hdf5_create_reference_scalar_to_2d (subroutine), 1305
io_hdf5_create_reference_scalar_to_3d (subroutine), 1305
io_hdf5_create_reference_scalar_to_4d (subroutine), 1305
io_hdf5_create_reference_scalar_to_5d (subroutine), 1305
io_hdf5_dataset_size (function), 1305
io_hdf5_destroy (subroutine), 1306
io_hdf5_has_attribute (function), 1306
io_hdf5_has_dataset (function), 1306
io_hdf5_has_group (function), 1306
io_hdf5_initialize (subroutine), 1306
io_hdf5_is_open (function), 1306
io_hdf5_is_reference (function), 1306
io_hdf5_open_attribute (function), 1306
io_hdf5_open_dataset (function), 1306
io_hdf5_open_file (subroutine), 1307
io_hdf5_open_group (function), 1307
io_hdf5_path_to (function), 1307
io_hdf5_read_attribute_character_1d_array_allocatable (subroutine), 1307
io_hdf5_read_attribute_character_1d_array_static (subroutine), 1307
io_hdf5_read_attribute_character_scalar (subroutine), 1307
io_hdf5_read_attribute_double_1d_array_allocatable (subroutine), 1307
io_hdf5_read_attribute_double_1d_array_static (subroutine), 1308
io_hdf5_read_attribute_double_scalar (subroutine), 1308
io_hdf5_read_attribute_integer8_1d_array_allocatable (subroutine), 1308
io_hdf5_read_attribute_integer8_1d_array_static (subroutine), 1308
io_hdf5_read_attribute_integer8_scalar (subroutine), 1308
io_hdf5_read_attribute_integer_1d_array_allocatable (subroutine), 1308
io_hdf5_read_attribute_integer_1d_array_static (subroutine), 1308
io_hdf5_read_attribute_integer_scalar (subroutine), 1308
io_hdf5_read_attribute_varstring_1d_array_allocatable (subroutine), 1309
io_hdf5_read_attribute_varstring_1d_array_allocatable_do_read (subroutine), 1309
io_hdf5_read_attribute_varstring_1d_array_static (subroutine), 1309
io_hdf5_read_attribute_varstring_1d_array_static_do_read (subroutine), 1309
io_hdf5_read_attribute_varstring_scalar (subroutine), 1309
io_hdf5_read_attribute_varstring_scalar_do_read (subroutine), 1309
io_hdf5_read_dataset_character_1d_array_allocatable (subroutine), 1309
io_hdf5_read_dataset_character_1d_array_static (subroutine), 1309
io_hdf5_read_dataset_double_1d_array_allocatable (subroutine), 1310
io_hdf5_read_dataset_double_1d_array_static (subroutine), 1310
io_hdf5_read_dataset_double_2d_array_allocatable (subroutine), 1310
io_hdf5_read_dataset_double_2d_array_static (subroutine), 1310
io_hdf5_read_dataset_double_3d_array_allocatable (subroutine), 1310
io_hdf5_read_dataset_double_3d_array_static (subroutine), 1310
io_hdf5_read_dataset_double_4d_array_allocatable (subroutine), 1310
io_hdf5_read_dataset_double_4d_array_static (subroutine), 1310
io_hdf5_read_dataset_double_5d_array_allocatable (subroutine), 1311
io_hdf5_read_dataset_double_5d_array_static (subroutine), 1311
io_hdf5_read_dataset_integer8_1d_array_allocatable (subroutine), 1311
io_hdf5_read_dataset_integer8_1d_array_static (subroutine), 1311
io_hdf5_read_dataset_integer_1d_array_allocatable (subroutine), 1311
io_hdf5_read_dataset_integer_1d_array_static (subroutine), 1311
io_hdf5_read_dataset_varstring_1d_array_allocatable (subroutine), 1311
io_hdf5_read_dataset_varstring_1d_array_allocatable_do_read (subroutine), 1311
io_hdf5_read_dataset_varstring_1d_array_static (subroutine), 1312
io_hdf5_read_dataset_varstring_1d_array_static_do_read (subroutine), 1312
io_hdf5_set_defaults (subroutine), 1312
io_hdf5_uninitialize (subroutine), 1312
io_hdf5_write_attribute_character_1d (subroutine), 1312
io_hdf5_write_attribute_character_scalar (subroutine), 1312
io_hdf5_write_attribute_double_1d (subroutine), 1312
io_hdf5_write_attribute_double_scalar (subroutine), 1312
io_hdf5_write_attribute_integer8_1d (subroutine), 1313
io_hdf5_write_attribute_integer8_scalar (subroutine), 1313
io_hdf5_write_attribute_integer_1d (subroutine), 1313
io_hdf5_write_attribute_integer_scalar (subroutine), 1313
io_hdf5_write_attribute_varstring_1d (subroutine), 1313
io_hdf5_write_attribute_varstring_scalar (subroutine), 1313
io_hdf5_write_dataset_character_1d (subroutine), 1313
io_hdf5_write_dataset_double_1d (subroutine), 1313
io_hdf5_write_dataset_double_2d (subroutine), 1314
io_hdf5_write_dataset_double_3d (subroutine), 1314
io_hdf5_write_dataset_double_4d (subroutine), 1314
io_hdf5_write_dataset_double_5d (subroutine), 1314
io_hdf5_write_dataset_integer8_1d (subroutine), 1314
io_hdf5_write_dataset_integer_1d (subroutine), 1314
io_hdf5_write_dataset_varstring_1d (subroutine), 1314
io_hdf_assert_is_initialized (subroutine), 1314
io_xml (module), 1315
ipmpar (function), 646
is_subhalo_subhalo_merger (function), 842
iso_varying_string (module), 801
iso_varying_string.F90 (file), 801
jeans_equation_integrand (function), 707
jeans_equation_integrand_einasto (function), 700
kepler_orbits (module), 903
kepler_orbits_angular_momentum (function), 903
kepler_orbits_angular_momentum_set (subroutine), 903
kepler_orbits_apocenter_radius (function), 903
kepler_orbits_apocenter_radius_set (subroutine), 903
kepler_orbits_assert_is_defined (subroutine), 903
kepler_orbits_builder (subroutine), 904
kepler_orbits_destroy (subroutine), 904
kepler_orbits_dump (subroutine), 904
kepler_orbits_dump_raw (subroutine), 904
kepler_orbits_eccentricity (function), 904
kepler_orbits_eccentricity_set (subroutine), 904
kepler_orbits_energy (function), 904
kepler_orbits_energy_set (subroutine), 904
kepler_orbits_host_mass (function), 904
kepler_orbits_is_bound (function), 905
kepler_orbits_is_defined (function), 905
kepler_orbits_masses_set (subroutine), 905
kepler_orbits_output (subroutine), 905
kepler_orbits_output_count (subroutine), 905
kepler_orbits_output_names (subroutine), 905
kepler_orbits_pericenter_radius (function), 905
kepler_orbits_pericenter_radius_set (subroutine), 905
kepler_orbits_propagate (subroutine), 905
kepler_orbits_radius (function), 906
kepler_orbits_radius_set (subroutine), 906
kepler_orbits_read_raw (subroutine), 906
kepler_orbits_reset (subroutine), 906
kepler_orbits_semi_major_axis (function), 906
kepler_orbits_semi_major_axis_set (subroutine), 906
kepler_orbits_specific_reduced_mass (function), 906
kepler_orbits_velocity_radial (function), 906
kepler_orbits_velocity_radial_set (subroutine), 907
kepler_orbits_velocity_tangential (function), 907
kepler_orbits_velocity_tangential_set (subroutine), 907
keplerorbit (type), 907
key_integer_scalar_i (function), 1322
keys_integer_scalar (subroutine), 1322
kind_numbers (module), 1337
kinetic_energy_integrand (function), 707
kinetic_energy_integrand_einasto (function), 700
kmt09_molecular_fraction_fast (function), 1208
kmt09_molecular_fraction_slow (function), 1208

last_host_descendent (function), 842
len (interface), 815
len_ (function), 815
len_trim (interface), 815
len_trim_ (function), 815
lge (interface), 815
lge_ch_vs (function), 815
lge_vs_ch (function), 815
lge_vs_vs (function), 815
lgt (interface), 815
lgt_ch_vs (function), 815
lgt_vs_ch (function), 816
lgt_vs_vs (function), 816
linear_growth (module), 1258
linear_growth_factor (function), 1259
linear_growth_factor_logarithmic_derivative (function), 1259
linear_growth_factor_simple_tabulate (subroutine), 1261
linear_growth_initialize (subroutine), 1259
linear_growth_simple (module), 1261
linear_growth_simple_state_retrieve (subroutine), 1261
linear_growth_simple_state_store (subroutine), 1261
linear_growth_state_retrieve (subroutine), 1259
little_h_0 (function), 685
lle (interface), 816
lle_ch_vs (function), 816
lle_vs_ch (function), 816
lle_vs_vs (function), 816
llt (interface), 816
llt_ch_vs (function), 816
llt_vs_ch (function), 816
llt_vs_vs (function), 816
load_from_file_vs (subroutine), 816
logarithmic_double_factorial (function), 825
luminositytable (type), 1226

make_distance_table (subroutine), 682
make_expansion_factor_table (subroutine), 679, 682
make_output_group (subroutine), 759
make_parameters_group (subroutine), 1337
make_range (function), 879
make_table (subroutine), 1272, 1273
make_variance_range (function), 1249
mapcomponentsdouble0 (function), 973
mapcomponentsvoid (subroutine), 973
mass_distribution_create (function), 908
mass_distribution_density_null (function), 908
mass_distribution_density_radial_moment_null (function), 908
mass_distribution_half_mass_radius_spherical (function), 908
mass_distribution_is_dimensionless (function), 908
mass_distribution_mass_enc_by_sphere_null (function), 908
mass_distribution_mass_enc_by_sphere_spherical (function), 908
mass_distribution_mass_enc_by_sphere_spherical_integrand (function), 908
mass_distribution_potential_null (function), 909
mass_distribution_symmetry_cylindrical (function), 909
mass_distribution_symmetry_spherical (function), 909
mass_distributions (module), 907
mass_from_cosmological_root_variance (function), 1263
mass_function_sampling_integrand (function), 836
massdistribution (type), 909
massdistributioncylindrical (type), 909
massdistributionspherical (type), 909
math.Bessel_functions.F90 (file), 823
math.distributions.Gaussian.F90 (file), 823
math.error_function.F90 (file), 824
math.exponential_integrals.F90 (file), 824
math.factorial.F90 (file), 825
math.gamma_function.F90 (file), 825
math.gamma_function.F90 (file), 825
math.gamma_function.F90 (file), 825
math.gamma_function.F90 (file), 825
math.gamma_function.F90 (file), 825
mean_density_contrast_root (function), 760
memory_management (module), 1339
memory_usage_get (function), 1353
memory_usage_record (subroutine), 1353
memory_usage_report (subroutine), 1353
memoryusage (type), 1353
memoryusagelist (type), 1353
merger_tree_branching (module), 827
merger_tree_branching_modifiers (module), 833
merger_tree_branching_modifiers_parkinson (function), 833
merger_tree_branching_modifiers_parkinson (function), 833
merger_tree_branching_modifiers_parkinson (function), 833
merger_tree_branching_modifiers_parkinson (function), 833
merger_tree_build (module), 835
merger_tree_build_cole2000 (module), 834
merger_tree_build_cole2000_snapshot (subroutine), 835
merger_tree_build_cole2000_snapshot (subroutine), 835
merger_tree_build_cole2000_state_retrieve (subroutine), 835
merger_tree_build_cole2000_state_store (subroutine), 835
merger_tree_build_do (subroutine), 836
merger_tree_build_do_cole2000 (subroutine), 835
merger_tree_build_initialize (subroutine), 836
merger_tree_cladistic_information_content (function), 914
merger_tree_constructFullySpecified (subroutine), 836
merger_tree_constructFullySpecifiedInitialize (subroutine), 837
merger_tree_construct_mass_function_sampling (function), 837
merger_tree_construct_mass_function_sampling_gaussian (function), 838
merger_tree_construct_mass_function_sampling_halo_mf (function), 838
merger_tree_construct_mass_function_sampling_power_law (function), 839
merger_tree_construct_mass_function_sampling_stellar_mf (function), 839
merger_tree_construction (module), 834
merger_tree_create (function), 834
merger_tree_data_construct_particle_indices (subroutine), 910
merger_tree_data_set_subhalo_masses (subroutine), 910
merger_tree_data_structure (module), 909
merger_tree_data_structure_add_metadata (subroutine), 910
merger_tree_data_structure_add_metadata_double (subroutine), 910
merger_tree_data_structure_add_metadata_integer (subroutine), 910
merger_tree_data_structure_add_metadata_text (subroutine), 910
merger_tree_data_structure_export (subroutine), 910
merger_tree_data_structure_export_galacticus (subroutine), 910
merger_tree_data_structure_export_irate (subroutine), 911
merger_tree_data_structure_make_references (subroutine), 911
merger_tree_data_structure_read_ascii (subroutine), 911
merger_tree_data_structure_read_particles_ascii (subroutine), 911
merger_tree_data_structure_reset (subroutine), 911
merger_tree_data_structure_setIncludesHubbleFlow (subroutine), 911
merger_tree_data_structure_setIncludesSubhaloMasses (subroutine), 911
merger_tree_data_structure_setIsPeriodic (subroutine), 911
merger_tree_data_structure_setNodeCount (subroutine), 912
merger_tree_data_structure_setParticleCount (subroutine), 912
merger_tree_data_structure_setParticleMass (subroutine), 912
merger_tree_data_structure_setParticlePropertyColumn (subroutine), 912
merger_tree_data_structure_setPropertyColumn (subroutine), 912
merger_tree_data_structure_setPropertyDouble (subroutine), 912
merger_tree_data_structure_setPropertyInteger8 (subroutine), 912
merger_tree_data_structure_setSelfContained (subroutine), 912
merger_tree_data_structure_setTreeCount (subroutine), 912
merger_tree_data_structure_setTreeIndices (subroutine), 913
merger_tree_data_structure_setUnits (subroutine), 913
merger_tree_data_validate_trees (subroutine), 913
merger_tree_dump (subroutine), 914
merger_tree_evolve_to (subroutine), 847
merger_tree_history_store (subroutine), 850
merger_tree_history_write (subroutine), 850
merger_tree_initialize (subroutine), 854
CODE INDEX

merger_tree_mass_accretion_history (module), 855
merger_tree_mass_accretion_history_close (subroutine), 855
merger_tree_mass_accretion_history_output (subroutine), 855
merger_tree_monotonic_mass_growth (subroutine), 855
merger_tree_output_structure (module), 855
merger_tree_prune_branches (subroutine), 856
merger_tree_prune_hierarchy (subroutine), 857
merger_tree_read (module), 840
merger_tree_read_do (subroutine), 842
merger_tree_read_initialize (subroutine), 843
merger_tree_read_state (module), 844
merger_tree_read_state_retrieve (subroutine), 844
merger_tree_read_state_store (subroutine), 844
merger_tree_record_evolution_output (subroutine), 851
merger_tree_record_evolution_store (subroutine), 851
merger_tree_regrid_time (subroutine), 857
merger_tree_smooth_accretion (module), 844
merger_tree_smooth_accretion_do (subroutine), 845
merger_tree_smooth_accretion_initialize (subroutine), 845
merger_tree_state_restore (subroutine), 845
merger_tree_state_store (subroutine), 845
merger_tree_state_store_initialize (subroutine), 845
merger_tree_state_walk_tree (subroutine), 846
merger_tree_structure_output (subroutine), 856
merger_tree_structure_preclose (subroutine), 856
merger_tree_timestep_history (subroutine), 850
merger_tree_timestep_record_evolution (subroutine), 851
merger_tree_timestep_satellite (subroutine), 852
merger_tree_timestep_simple (subroutine), 853
merger_tree_timesteps (module), 849
merger_tree_timesteps_history (module), 850
merger_tree_timesteps_record_evolution (module), 851
merger_tree_timesteps_satellite (module), 852
merger_tree_timesteps_simple (module), 853
merger_tree_walk_descend_to_progenitors (function), 973
merger_tree_write (subroutine), 858
merger_trees.branching_probability.F90 (file), 827
merger_trees.branching_probability.generalized_Press_Schechter.F90 (file), 829
merger_trees.branching_probability.modified_Press_Schechter.F90 (file), 831
merger_trees.branching_probability.modifier.F90 (file), 832
merger_trees.branching_probability.modifier.null.F90 (file), 833
merger_trees.branching_probability.modifier.Parkinson.F90 (file), 833
merger_trees.construct.build.Cole2000.F90 (file), 834
merger_trees.construct.build.F90 (file), 835
merger_trees.construct.F90 (file), 834
merger_trees.construct.fully_specified.F90 (file), 836
merger_trees.construct.mass_function_sampling.F90 (file), 837
merger_trees.construct.mass_function_sampling.gaussian.F90 (file), 838
merger_trees.construct.mass_function_sampling.halo_mass_function.F90 (file), 838
merger_trees.construct.mass_function_sampling.power_law.F90 (file), 839
merger_trees.construct.mass_function_sampling.stellar_mass_function.F90 (file), 839
merger_trees.construct.read.F90 (file), 840
merger_trees.construct.read.state.F90 (file), 844
merger_trees.construct.smooth_accretion.F90 (file), 844
merger_trees.construct.state_restore.F90 (file), 845
merger_trees.evolve.deadlock_options.F90 (file), 847
merger_trees.evolve.F90 (file), 846
merger_trees.evolve.node.F90 (file), 847
merger_trees.evolve.timesteps.F90 (file), 849
merger_trees.evolve.timesteps.history.F90 (file), 850
merger_trees.evolve.timesteps.record_evolution.F90 (file), 851
merger_trees.evolve.timesteps.report.F90 (file), 851
merger_trees.evolve.timesteps.satellite.F90 (file), 852
merger_trees.evolve.timesteps.simple.F90 (file), 853
merger_trees.evolve.timesteps.template.F90 (file), 853
merger_trees.file_maker.Millennium.F90 (file), 853
merger_trees.file_maker.simple.F90 (file), 854
merger_trees.initialize.F90 (file), 854
merger_trees.mass_accretion_history.F90 (file), 855
merger_trees.monotonic_mass_growth.F90 (file), 855
merger_trees.output_structure.F90 (file), 855
merger_trees.prune_branches.F90 (file), 856
merger_trees.prune_hierarchy.F90 (file), 856
merger_trees.regrid_times.F90 (file), 857
merger_trees.render.F90 (file), 857
merger_trees.write.F90 (file), 858
merger_trees_construct_fully_specified (module), 836
merger_trees_dump (module), 914
merger_trees_evolve (module), 846
merger_trees_evolve_deadlock_status (module), 847
merger_trees_evolve_node (module), 847
merger_trees_evolve_timesteps_template (module), 853
merger_trees_information_content (module), 914
merger_trees_initialize (module), 854
merger_trees_mass_function_sampling (module), 837
merger_trees_mass_function_sampling_gaussian (module), 838
merger_trees_mass_function_sampling_gaussian_initialize (subroutine), 838
merger_trees_mass_function_sampling_halo_mf (module), 838
merger_trees_mass_function_sampling_halo_mf_initialize (subroutine), 838
merger_trees_mass_function_sampling_initialize (subroutine), 837
merger_trees_mass_function_sampling_power_law (module), 839
merger_trees_mass_function_sampling_power_law_initialize (subroutine), 839
merger_trees_mass_function_sampling_stellar_mf (module), 839
merger_trees_mass_function_sampling_stellar_mf_initialize (subroutine), 839
merger_trees_millennium (module), 853
merger_trees_millennium_process (subroutine), 853
merger_trees_monotonic_mass_growth (module), 855
merger_trees_prune_branches (module), 856
merger_trees_prune_hierarchy (module), 856
merger_trees_regrid_times (module), 857
merger_trees_render (module), 857
merger_trees_render_dump (subroutine), 857
merger_trees_simple (module), 854
merger_trees_simple_process (subroutine), 854
merger_trees_state_store (module), 845
merger_trees_write (module), 858
mergertreedata (type), 913
merging_rate (function), 830, 831
mergingstatisticscountlinked (function), 973
mergingstatisticscreatebyinterrupt (subroutine), 973
mergingstatisticscreatelinked (subroutine), 973
mergingstatisticsdestroylinked (subroutine), 974
mergingstatisticsgalaxymajormergertime (function), 974
mergingstatisticsgalaxymajormergertimeisgettable (function), 974
mergingstatisticsget (function), 974
mergingstatisticsnodeformationtime (function), 974
mergingstatisticsnodeformationtimeisgettable (function), 974
mergingstatisticsnodehierarchylevel (function), 974
mergingstatisticsnodehierarchylevelisgettable (function), 974
mergingstatisticsnodemajormergertime (function), 975
mergingstatisticsnodemajormergertimeisgettable (function), 975
mergingstatisticsnullbindingdouble0inout (function), 975
mergingstatisticsnullbindinginteger0in (function), 975
mergingstatisticsnullbindinginteger0inout (function), 975
mergingstatisticsnullbindinginteger1inout (function), 975
mergingstatisticsnullbindingratedouble0inout (subroutine), 975
mergingstatisticsnullbindingrateinteger0in (subroutine), 975
mergingstatisticsnullbindingrateinteger0inout (subroutine), 975
mergingstatisticsnullbindingrateinteger1inout (subroutine), 975
mergingstatisticsnullbindingsetdouble0inout (subroutine), 976
mergingstatisticsnullbindingsetinteger0in (subroutine), 976
mergingstatisticsnullbindingsetinteger0inout (subroutine), 976
mergingstatisticsnullbindingsetinteger1inout (subroutine), 976
mergingstatisticsrecentmajormergercount (function), 976
mergingstatisticsrecentmajormergercountisgettable (function), 976
mergingstatisticsrecentrecentmajormergercountget (function), 976
mergingstatisticsrecentrecentmajormergercountset (subroutine), 976
mergingstatisticsstandardgalaxymajormergertimeget (function), 977
mergingstatisticsstandardgalaxymajormergertimeisgettable (function), 977
mergingstatisticsstandardnodeformationtimeget (function), 977
mergingstatisticsstandardnodeformationtimeisgettable (function), 977
mergingstatisticsstandardnodehierarchylevelget (function), 977
mergingstatisticsstandardnodehierarchylevelisgettable (function), 977
mergingstatisticsstandardnodemajormergertimeget (function), 977
mergingstatisticsstandardnodemajormergertimeisgettable (function), 977
meshes (module), 876
meshes_apply_point (subroutine), 876
CODE INDEX

meta_tree_timing_initialize (subroutine), 754
meta_tree_timing_output (subroutine), 754
meta_tree_timing_post_evolve (subroutine), 755
meta_tree_timing_pre_construction (subroutine), 755
meta_tree_timing_pre_evolve (subroutine), 755
metal_yield_integrand (function), 1187
millennium_merger_tree_file_maker (program), 600
Millennium_Merger_Tree_File_Maker.F90 (file), 600
modification_function (function), 831
modified_press_schechter_branch_mass (function), 831
modified_press_schechter_branch_mass_root (function), 832
modified_press_schechter_branching (module), 831
modified_press_schechter_branching_initialize (subroutine), 832
modified_press_schechter_branching_maximum_step (function), 832
modified_press_schechter_branching_probability (function), 832
modified_press_schechter_subresolution_fraction (function), 832
nfw_concentration_function_root (function), 713
nfw_profile_energy (function), 707
node_array_position (function), 846
node_branch_jump (function), 717
node_branch_jumps (module), 717
node_component_assign (subroutine), 978
node_component_basic_builder (subroutine), 978
node_component_basic_destroy (subroutine), 978
node_component_basic_dump (subroutine), 978
node_component_basic_initializor (subroutine), 978
node_component_basic_move (subroutine), 978
node_component_basic_non_evolving (module), 1118
node_component_basic_non_evolving_promote (subroutine), 1118
node_component_basic_non_evolving_rate_compute (subroutine), 1119
node_component_basic_non_evolving_scale_set (subroutine), 1119
node_component_basic_output (subroutine), 978
node_component_basic_output_count (subroutine), 978
node_component_basic_output_names (subroutine), 978
node_component_basic_remove (subroutine), 979
node_component_basic_standard (module), 1119
node_component_basic_standard_promote (subroutine), 1119
node_component_basic_standard_rate_compute (subroutine), 1119
node_component_basic_standard_scale_set (subroutine), 1119
node_component_basic_standard_stop_accretion (subroutine), 1119
node_component_basic_standard_tree_initialize (subroutine), 1120
node_component_basic_standard_unresolved_mass (function), 1120
node_component_basic_type (function), 979
node_component_basicnon evolving_builder (subroutine), 979
node_component_basicnon evolving_count (function), 979
node_component_basicnon evolving_deserialize_rates (subroutine), 979
node_component_basicnon evolving_deserialize_scales (subroutine), 979
node_component_basicnon evolving_deserialize_values (subroutine), 979
node_component_basicnon evolving_destroy (subroutine), 979
node_component_basicnonevolving_dump (subroutine), 980
node_component_basicnonevolving_dump_raw (subroutine), 980
node_component_basicnonevolving_initializer (subroutine), 980
node_component_basicnonevolving_is_active (function), 980
node_component_basicnonevolving_name_from_index (subroutine), 980
node_component_basicnonevolving_ode_step_rates_init (subroutine), 980
node_component_basicnonevolving_ode_step_scales_init (subroutine), 980
node_component_basicnonevolving_output_count (subroutine), 980
node_component_basicnonevolving_output_names (subroutine), 980
node_component_basicnonevolving_read_raw (subroutine), 981
node_component_basicnonevolving_serialize_rates (subroutine), 981
node_component_basicnonevolving_serialize_scales (subroutine), 981
node_component_basicnonevolving_serialize_values (subroutine), 981
node_component_basicnonevolving_type (function), 981
node_component_basicnull_builder (subroutine), 981
node_component_basicnull_count (function), 981
node_component_basicnull_deserialize_rates (subroutine), 981
node_component_basicnull_deserialize_scales (subroutine), 981
node_component_basicnull_deserialize_values (subroutine), 982
node_component_basicnull_destroy (subroutine), 982
node_component_basicnull_dump (subroutine), 982
node_component_basicnull_dump_raw (subroutine), 982
node_component_basicnull_initializer (subroutine), 982
node_component_basicnull_is_active (function), 982
node_component_basicnull_name_from_index (subroutine), 982
node_component_basicnull_ode_step_rates_init (subroutine), 982
node_component_basicnull_ode_step_scales_init (subroutine), 983
node_component_basicnull_output_count (subroutine), 983
node_component_basicnull_output_names (subroutine), 983
node_component_basicnull_read_raw (subroutine), 983
node_component_basicnull_serialize_rates (subroutine), 983
node_component_basicnull_serialize_scales (subroutine), 983
node_component_basicnull_serialize_values (subroutine), 983
node_component_basicnull_type (function), 983
node_component_basicstandard_builder (subroutine), 983
node_component_basicstandard_count (function), 984
node_component_basicstandard_deserialize_rates (subroutine), 984
node_component_basicstandard_deserialize_scales (subroutine), 984
node_component_basicstandard_deserialize_values (subroutine), 984
node_component_basicstandard_destroy (subroutine), 984
node_component_basicstandard_dump (subroutine), 984
node_component_basicstandard_dump_raw (subroutine), 984
node_component_basicstandard_initializer (subroutine), 984
node_component_basicstandard_is_active (function), 984
node_component_basicstandard_name_from_index (subroutine), 985
node_component_basicstandard_ode_step_rates_init (subroutine), 985
node_component_basicstandard_ode_step_scales_init (subroutine), 985
node_component_basicstandard_output_count (subroutine), 985
node_component_basicstandard_output_names (subroutine), 985
node_component_basicstandard_read_raw (subroutine), 985
node_component_basicstandard_serialize_rates (subroutine), 985
node_component_basicstandard_serialize_scales (subroutine), 985
node_component_basicstandard_serialize_values (subroutine), 985
node_component_basicstandard_type (function), 986
node_component_black_hole_simple (module), 1120
node_component_black_hole_simple_create (subroutine), 1120
node_component_black_hole_simple_initialize (subroutine), 1120
node_component_black_hole_simple_matches (function), 1120
node_component_black_hole_simple_output (subroutine), 1120
node_component_black_hole_simple_output_count (subroutine), 1121
node_component_black_hole_simple_output_names (subroutine), 1121
node_component_black_hole_simple_potential (function), 1121
node_component_black_hole_simple_rate_compute (subroutine), 1121
node_component_black_hole_simple_rotation_curve (function), 1122
node_component_black_hole_simple_rotation_curve_gradient (function), 1122
node_component_black_hole_simple_satellite_merging (subroutine), 1121
node_component_black_hole_simple_scale_set (subroutine), 1121
node_component_black_hole_simple_structure (module), 1121
node_component_black_hole_standard (module), 1122
node_component_black_hole_standard_create (subroutine), 1122
node_component_black_hole_standard_initialize (subroutine), 1123
node_component_black_hole_standard_mass_accretion_rate (subroutine), 1123
node_component_black_hole_standard_matches (function), 1123
node_component_black_hole_standard_merge_black_holes (subroutine), 1123
node_component_black_hole_standard_output (subroutine), 1123
node_component_black_hole_standard_output_count (subroutine), 1123
node_component_black_hole_standard_output_merge (subroutine), 1123
node_component_black_hole_standard_output_names (subroutine), 1123
node_component_black_hole_standard_output_properties (subroutine), 1124
node_component_black_hole_standard_potential (function), 1125
node_component_black_hole_standard_rate_compute (subroutine), 1124
node_component_black_hole_standard_rotation_curve (function), 1125
node_component_black_hole_standard_rotation_curve_gradient (function), 1125
node_component_black_hole_standard_satellite_merging (subroutine), 1124
node_component_black_hole_standard_scale_set (subroutine), 1124
node_component_black_hole_standard_structure_tasks (module), 1124
node_component_black_hole_standard_triple_interaction (subroutine), 1124
node_component_blackhole_builder (subroutine), 986
node_component_blackhole_destroy (subroutine), 986
node_component_blackhole_dump (subroutine), 986
node_component_blackhole_initializor (subroutine), 986
node_component_blackhole_move (subroutine), 986
node_component_blackhole_output (subroutine), 986
node_component_blackhole_output_count (subroutine), 986
node_component_blackhole_output_names (subroutine), 986
node_component_blackhole_remove (subroutine), 987
node_component_blackhole_type (function), 987
node_component_blackholenull_builder (subroutine), 987
node_component_blackholenull_count (function), 987
node_component_blackholenull_deserialize_rates (subroutine), 987
node_component_blackholenull_deserialize_scales (subroutine), 987
node_component_blackholenull_deserialize_values (subroutine), 987
node_component_blackholenull_destroy (subroutine), 987
node_component_blackholenull_dump (subroutine), 987
node_component_blackholenull_dump_raw (subroutine), 988
node_component_blackholenull_initializor (subroutine), 988
node_component_blackholenull_is_active (function), 988
node_component_blackholenull_name_from_index (subroutine), 988
node_component_blackholenull_ode_step_rates_init (subroutine), 988
node_component_blackholenull_ode_step_scales_init (subroutine), 988
node_component_blackholenull_output_count (subroutine), 988
node_component_blackholenull_output_names (subroutine), 988
node_component_blackholenull_read_raw (subroutine), 988
node_component_blackholenull_serialize_rates (subroutine), 989
node_component_blackholenull_serialize_scales (subroutine), 989
node_component_blackholenull_serialize_values (subroutine), 989
node_component_blackholenull_type (function), 989
node_component_blackholesimple_builder (subroutine), 989
node_component_blackholesimple_count (function), 989
node_component_blackholesimple_deserialize_rates (subroutine), 989
node_component_blackholesimple_deserialize_scales (subroutine), 989
node_component_blackholesimple_deserialize_values (subroutine), 989
node_component_blackholesimple_destroy (subroutine), 990
node_component_blackholesimple_dump (subroutine), 990
node_component_blackholesimple_dump_raw (subroutine), 990
node_component_blackholesimple_initializor (subroutine), 990
node_component_blackholesimple_is_active (function), 990
node_component_blackholesimple_name_from_index (subroutine), 990
node_component_blackholesimple_ode_step_rates_init (subroutine), 990
node_component_blackholesimple_ode_step_scales_init (subroutine), 990
node_component_blackholesimple_output_count (subroutine), 991
node_component_blackholesimple_output_names (subroutine), 991
node_component_blackholesimple_read_raw (subroutine), 991
node_component_blackholesimple_serialize_rates (subroutine), 991
node_component_blackholesimple_serialize_scales (subroutine), 991
node_component_blackholesimple_serialize_values (subroutine), 991
node_component_blackholesimple_type (function), 991
node_component_blackholestandard_builder (subroutine), 991
node_component_blackholestandard_count (function), 991
node_component_blackholestandard_deserialize_rates (subroutine), 992
node_component_blackholestandard_deserialize_scales (subroutine), 992
node_component_blackholestandard_deserialize_values (subroutine), 992
node_component_blackholestandard_destroy (subroutine), 992
node_component_blackholestandard_dump (subroutine), 992
node_component_blackholestandard_dump_raw (subroutine), 992
node_component_blackholestandard_initializor (subroutine), 992
node_component_blackholestandard_is_active (function), 992
node_component_blackholestandard_name_from_index (subroutine), 992
node_component_blackholestandard_ode_step_rates_init (subroutine), 993
node_component_blackholestandard_ode_step_scales_init (subroutine), 993
node_component_blackholestandard_output_count (subroutine), 993
node_component_blackholestandard_output_names (subroutine), 993
node_component_blackholestandard_read_raw (subroutine), 993
node_component_blackholestandard_serialize_rates (subroutine), 993
node_component_blackholestandard_serialize_scales (subroutine), 993
node_component_blackholestandard_serialize_values (subroutine), 993
node_component_blackholestandard_type (function), 993
node_component_dark_matter_profile_scale (module), 1125
node_component_dark_matter_profile_scale_initialize (subroutine), 1126
node_component_dark_matter_profile_scale_initialize_scale (subroutine), 1126
node_component_dark_matter_profile_scale_plausibility (subroutine), 1126
node_component_dark_matter_profile_scale_promote (subroutine), 1126
node_component_dark_matter_profile_scale_rate_compute (subroutine), 1126
node_component_dark_matter_profile_scale_scale (function), 1126
node_component_dark_matter_profile_scale_scale_set (subroutine), 1126
node_component_dark_matter_profile_scale_scale_shape_set (subroutine), 1126
node_component_dark_matter_profile_scale_shape (module), 1127
node_component_dark_matter_profile_scale_shape_initialize (subroutine), 1127
node_component_dark_matter_profile_scale_shape_initialize_shape (subroutine), 1127
node_component_dark_matter_profile_scale_shape_promote (subroutine), 1127
node_component_dark_matter_profile_scale_shape_rate_compute (subroutine), 1127
node_component_dark_matter_profile_scale_shape_scale_set (subroutine), 1127
node_component_dark_matter_profile_scale_shape_tree_initialize (subroutine), 1128
node_component_dark_matter_profile_scale_shape_tree_output (subroutine), 1128
node_component_dark_matter_profile_scale_shape_tree_output (subroutine), 1128
node_component_dark_matter_profile_scale_tree_initialize (subroutine), 1126
node_component_darkmatterprofile_builder (subroutine), 994
node_component_darkmatterprofile_destroy (subroutine), 994
node_component_darkmatterprofile_dump (subroutine), 994
node_component_darkmatterprofile_initializor (subroutine), 994
node_component_darkmatterprofile_move (subroutine), 994
node_component_darkmatterprofile_output (subroutine), 994
node_component_darkmatterprofile_output_count (subroutine), 994
node_component_darkmatterprofile_output_names (subroutine), 994
node_component_darkmatterprofile_remove (subroutine), 994
node_component_darkmatterprofile_type (function), 995
node_component_darkmatterprofilenull_builder (subroutine), 995
node_component_darkmatterprofilenull_count (function), 995
node_component_darkmatterprofilenull_deserialize_rates (subroutine), 995
node_component_darkmatterprofilenull_deserialize_scales (subroutine), 995
node_component_darkmatterprofilenull_deserialize_values (subroutine), 995
node_component_darkmatterprofilenull_destroy (subroutine), 995
node_component_darkmatterprofilenull_dump (subroutine), 995
node_component_darkmatterprofilenull_dump_raw (subroutine), 996
node_component_darkmatterprofilenull_initializor (subroutine), 996
node_component_darkmatterprofilenull_is_active (function), 996
node_component_darkmatterprofilenull_name_from_index (subroutine), 996
node_component_darkmatterprofilenull_ode_step_rates_init (subroutine), 996
node_component_darkmatterprofilenull_ode_step_scales_init (subroutine), 996
node_component_darkmatterprofilenull_output_count (subroutine), 996
node_component_darkmatterprofilenull_output_names (subroutine), 996
node_component_darkmatterprofilenull_read_raw (subroutine), 996
node_component_darkmatterprofilenull_serialize_rates (subroutine), 997
node_component_darkmatterprofilenull_serialize_scales (subroutine), 997
node_component_darkmatterprofilenull_serialize_values (subroutine), 997
node_component_darkmatterprofilenull_type (function), 997
node_component_darkmatterprofilescale_builder (subroutine), 997
node_component_darkmatterprofilescale_count (function), 997
node_component_darkmatterprofilescale_deserialize_rates (subroutine), 997
node_component_darkmatterprofilescale_deserialize_scales (subroutine), 997
node_component_darkmatterprofilescale_deserialize_values (subroutine), 997
node_component_darkmatterprofilescale_destroy (subroutine), 998
node_component_darkmatterprofilescale_dump (subroutine), 998
node_component_darkmatterprofilescale_dump_raw (subroutine), 998
node_component_darkmatterprofilescale_initializor (subroutine), 998
node_component_darkmatterprofilescale_is_active (function), 998
node_component_darkmatterprofilescale_name_from_index (subroutine), 998
node_component_darkmatterprofilescale_ode_step_rates_init (subroutine), 998
node_component_darkmatterprofilescale_ode_step_scales_init (subroutine), 998
node_component_darkmatterprofilescale_output_count (subroutine), 999
node_component_darkmatterprofilescale_output_names (subroutine), 999
node_component_darkmatterprofilescale_read_raw (subroutine), 999
node_component_darkmatterprofilescale_serialize_rates (subroutine), 999
node_component_darkmatterprofilescale_serialize_scales (subroutine), 999
node_component_darkmatterprofilescale_serialize_values (subroutine), 999
node_component_darkmatterprofilescale_type (function), 999
node_component_darkmatterprofileshape_builder (subroutine), 999
node_component_darkmatterprofileshape_count (function), 999
node_component_darkmatterprofileshape_deserialize_rates (subroutine), 1000
node_component_darkmatterprofileshape_deserialize_scales (subroutine), 1000
node_component_darkmatterprofileshape_deserialize_values (subroutine), 1000
node_component_darkmatterprofileshape_destroy (subroutine), 1000
node_component_darkmatterprofileshape_dump (subroutine), 1000
node_component_darkmatterprofileshape_dump_raw (subroutine), 1000
node_component_darkmatterprofileshape_initializor (subroutine), 1000
node_component_darkmatterprofileshape_is_active (function), 1000
node_component_darkmatterprofileshape_name_from_index (subroutine), 1000
node_component_darkmatterprofileshape_ode_step_rates_init (subroutine), 1001
node_component_darkmatterprofileshape_ode_step_scales_init (subroutine), 1001
node_component_darkmatterprofileshape_output_count (subroutine), 1001
node_component_darkmatterprofileshape_output_names (subroutine), 1001
node_component_darkmatterprofileshape_read_raw (subroutine), 1001
node_component_darkmatterprofileshape_serialize_rates (subroutine), 1001
node_component_darkmatterprofileshape_serialize_scales (subroutine), 1001
node_component_darkmatterprofileshape_serialize_values (subroutine), 1001
node_component_darkmatterprofileshape_type (function), 1002
node_component_density_null (function), 1002
node_component_deserialize_null (subroutine), 1002
node_component_disk_builder (subroutine), 1002
node_component_disk_destroy (subroutine), 1002
node_component_disk_dump (subroutine), 1002
node_component_disk_exponential (module), 1128
node_component_disk_exponential_calculation_reset (subroutine), 1128
node_component_disk_exponential_create (subroutine), 1128
node_component_disk_exponential_data (module), 1131
node_component_disk_exponential_enclosed_mass_dimensionless (function), 1131
node_component_disk_exponential_initialize (subroutine), 1128
node_component_disk_exponential_post_evolve (subroutine), 1129
node_component_disk_exponential_pre_evolve (subroutine), 1129
node_component_disk_exponential_radius_solve (function), 1129
node_component_disk_exponential_radius_solve_set (subroutine), 1129
node_component_disk_exponential_radius_solver (subroutine), 1129
node_component_disk_exponential_radius_solver_plausibility (subroutine), 1129
node_component_disk_exponential_rate_compute (subroutine), 1129
node_component_disk_exponential_reset (subroutine), 1131
node_component_disk_exponential_rotation_curve_bessel_factors (function), 1131
node_component_disk_exponential_satellite_merging (subroutine), 1130
node_component_disk_exponential_scale_set (subroutine), 1130
node_component_disk_exponential_star_formation_history_output (subroutine), 1130
node_component_disk_exponential_star_formation_rate (function), 1130
node_component_disk_exponential_state_retrieve (subroutine), 1131
node_component_disk_exponential_state_store (subroutine), 1131
node_component_disk_exponential_velocity (function), 1130
node_component_disk_exponential_velocity_set (subroutine), 1130
node_component_disk_initializer (subroutine), 1002
node_component_disk_move (subroutine), 1002
node_component_disk_output (subroutine), 1002
node_component_disk_output_count (subroutine), 1003
node_component_disk_output_names (subroutine), 1003
node_component_disk_remove (subroutine), 1003
node_component_disk_type (function), 1003
node_component_disk_very_simple (module), 1132
node_component_disk_very_simple_initialize (subroutine), 1132
node_component_disk_very_simple_post_evolve (subroutine), 1132
node_component_disk_very_simple_rate_compute (subroutine), 1132
node_component_disk_very_simple_satellite_merging (subroutine), 1132
node_component_disk_very_simple_scale_set (subroutine), 1132
node_component_disk_very_simple_sfr (function), 1132
node_component_disk_exponential_builder (subroutine), 1003
node_component_disk_exponential_count (function), 1003
node_component_disk_exponential_deserialize_rates (subroutine), 1003
node_component_disk_exponential_deserialize_scales (subroutine), 1003
node_component_disk_exponential_deserialize_values (subroutine), 1003
node_component_disk_exponential_destroy (subroutine), 1004
node_component_diskexponential_dump (subroutine), 1004
node_component_diskexponential_dump_raw (subroutine), 1004
node_component_diskexponential_initializor (subroutine), 1004
node_component_diskexponential_is_active (function), 1004
node_component_diskexponential_name_from_index (subroutine), 1004
node_component_diskexponential_ode_step_rates_init (subroutine), 1004
node_component_diskexponential_ode_step_scales_init (subroutine), 1004
node_component_diskexponential_output_count (subroutine), 1005
node_component_diskexponential_output_names (subroutine), 1005
node_component_diskexponential_read_raw (subroutine), 1005
node_component_diskexponential_serialize_rates (subroutine), 1005
node_component_diskexponential_serialize_scales (subroutine), 1005
node_component_diskexponential_serialize_values (subroutine), 1005
node_component_diskexponential_type (function), 1005
node_component_disknull_builder (subroutine), 1005
node_component_disknull_count (function), 1006
node_component_disknull_deserialize_rates (subroutine), 1006
node_component_disknull_deserialize_scales (subroutine), 1006
node_component_disknull_deserialize_values (subroutine), 1006
node_component_disknull_destroy (subroutine), 1006
node_component_disknull_dump (subroutine), 1006
node_component_disknull_dump_raw (subroutine), 1006
node_component_disknull_initializer (subroutine), 1006
node_component_disknull_is_active (function), 1006
node_component_disknull_name_from_index (subroutine), 1007
node_component_disknull_ode_step_rates_init (subroutine), 1007
node_component_disknull_ode_step_scales_init (subroutine), 1007
node_component_disknull_output_count (subroutine), 1007
node_component_disknull_output_names (subroutine), 1007
node_component_disknull_read_raw (subroutine), 1007
node_component_disknull_serialize_rates (subroutine), 1007
node_component_disknull_serialize_scales (subroutine), 1007
node_component_disknull_serialize_values (subroutine), 1007
node_component_disknull_type (function), 1008
node_component_diskverysimple_builder (subroutine), 1008
node_component_diskverysimple_count (function), 1008
node_component_diskverysimple_deserialize_rates (subroutine), 1008
node_component_diskverysimple_deserialize_scales (subroutine), 1008
node_component_diskverysimple_deserialize_values (subroutine), 1008
node_component_diskverysimple_destroy (subroutine), 1008
node_component_diskverysimple_dump (subroutine), 1008
node_component_diskverysimple_dump_raw (subroutine), 1008
node_component_diskverysimple_initializor (subroutine), 1009
node_component_diskverysimple_is_active (function), 1009
node_component_diskverysimple_name_from_index (subroutine), 1009
node_component_diskverysimple_ode_step_rates_init (subroutine), 1009
node_component_diskverysimple_ode_step_scales_init (subroutine), 1009
node_component_diskverysimple_output_count (subroutine), 1009
node_component_diskverysimple_output_names (subroutine), 1009
node_component_diskverysimple_read_raw (subroutine), 1009
node_component_diskverysimple_serialize_rates (subroutine), 1009
node_component_diskverysimpleSerialize_scales (subroutine), 1010
node_component_diskverysimple_serialize_values (subroutine), 1010
node_component_diskverysimple_type (function), 1010
node_component_dump_null (subroutine), 1010
node_component_dump_raw_null (subroutine), 1010
node_component_enclosed_mass_null (function), 1010
node_component_formation_time_cole2000_create (subroutine), 1133
node_component_formation_time_cole2000_node_promotion (subroutine), 1133
node_component_formation_time_cole2000_rate_compute (subroutine), 1133
node_component_formation_time_cole2000_tree_initialize (subroutine), 1133
node_component_formation_times_cole2000 (module), 1133
node_component_formation_times_cole2000_initialize (subroutine), 1133
node_component_formationtime_builder (subroutine), 1010
node_component_formationtime_destroy (subroutine), 1010
node_component_formationtime_dump (subroutine), 1010
node_component_formationtime_initializor (subroutine), 1011
node_component_formationtime_move (subroutine), 1011
node_component_formationtime_output (subroutine), 1011
node_component_formationtime_output_count (subroutine), 1011
node_component_formationtime_output_names (subroutine), 1011
node_component_formationtime_remove (subroutine), 1011
node_component_formationtime_type (function), 1011
node_component_formationtimecole2000_builder (subroutine), 1011
node_component_formationtimecole2000_count (function), 1012
node_component_formationtimecole2000_deserialize_rates (subroutine), 1012
node_component_formationtimecole2000_deserialize_scales (subroutine), 1012
node_component_formationtimecole2000_deserialize_values (subroutine), 1012
node_component_formationtimecole2000_destroy (subroutine), 1012
node_component_formationtimecole2000_dump (subroutine), 1012
node_component_formationtimecole2000_dump_raw (subroutine), 1012
node_component_formationtimecole2000_initializor (subroutine), 1012
node_component_formationtimecole2000_is_active (function), 1012
node_component_formationtimecole2000_name_from_index (subroutine), 1013
node_component_formationtimecole2000_ode_step_rates_init (subroutine), 1013
node_component_formationtimecole2000_ode_step_scales_init (subroutine), 1013
node_component_formationtimecole2000_output_count (subroutine), 1013
node_component_formationtimecole2000_output_names (subroutine), 1013
node_component_formationtimecole2000_read_raw (subroutine), 1013
node_component_formationtimecole2000_serialize_rates (subroutine), 1013
node_component_formationtimecole2000_serialize_scales (subroutine), 1013
node_component_formationtimecole2000_serialize_values (subroutine), 1013
node_component_formationtimecole2000_type (function), 1014
node_component_formationtimenull_builder (subroutine), 1014
node_component_formationtimenull_count (function), 1014
node_component_formationtimenull_deserialize_rates (subroutine), 1014
node_component_formationtimenull_deserialize_scales (subroutine), 1014
node_component_formationtimenull_deserialize_values (subroutine), 1014
node_component_formationtimenull_destroy (subroutine), 1014
node_component_formationtimenull_dump (subroutine), 1014
node_component_formationtimenull_dump_raw (subroutine), 1015
node_component_formationtimenull_initializer (subroutine), 1015
node_component_formationtimenull_is_active (function), 1015
node_component_formationtimenull_name_from_index (subroutine), 1015
node_component_formationtimenull_ode_step_rates_init (subroutine), 1015
node_component_formationtimenull_ode_step_scales_init (subroutine), 1015
node_component_formationtimenull_output_count (subroutine), 1015
node_component_formationtimenull_output_names (subroutine), 1015
node_component_formationtimenull_read_raw (subroutine), 1015
node_component_formationtimenull_serialize_rates (subroutine), 1016
node_component_formationtimenull_serialize_scales (subroutine), 1016
node_component_formationtimenull_serialize_values (subroutine), 1016
node_component_formationtimenull_type (function), 1016
node_component_generic_destroy (subroutine), 1016
node_component_generic_type (function), 1016
node_component_host_node (function), 1016
node_component_hot_halo_standard (module), 1133
node_component_hot_halo_standard_cooling_rate (subroutine), 1134
node_component_hot_halo_standard_create (subroutine), 1134
node_component_hot_halo_standard_formation (subroutine), 1134
node_component_hot_halo_standard_heat_source (subroutine), 1134
node_component_hot_halo_standard_initialize (subroutine), 1134
node_component_hot_halo_standard_node_merger (subroutine), 1134
node_component_hot_halo_standard_outer_radius (function), 1135
node_component_hot_halo_standard_outflow_stripped_fraction (function), 1135
node_component_hot_halo_standard_outflowing_abundances_rate (subroutine), 1135
node_component_hot_halo_standard_outflowing_ang_mom_rate (subroutine), 1135
node_component_hot_halo_standard_outflowing_mass_rate (subroutine), 1135
node_component_hot_halo_standard_post_evolve (subroutine), 1135
node_component_hot_halo_standard_promote (subroutine), 1135
node_component_hot_halo_standard_push_to_cooling_pipes (subroutine), 1135
node_component_hot_halo_standard_push_to_null (subroutine), 1136
node_component_hot_halo_standard_rate_compute (subroutine), 1136
node_component_hot_halo_standard_reset (subroutine), 1136
node_component_hot_halo_standard_satellite_merger (subroutine), 1136
node_component_hot_halo_standard_scale_set (subroutine), 1136
node_component_hot_halo_standard_strip_gas_rate (subroutine), 1136
node_component_hot_halo_standard_thread_initialize (subroutine), 1137
node_component_hot_halo_standard_tree_initialize (subroutine), 1137
node_component_hot_halo_variable_simple (module), 1137
node_component_hot_halo_variable_simple_cooling_rate (subroutine), 1137
node_component_hot_halo_variable_simple_create (subroutine), 1137
node_component_hot_halo_variable_simple_initialize (subroutine), 1137
node_component_hot_halo_variable_simple_node_merger (subroutine), 1137
node_component_hot_halo_variable_simple_outflowing_mass_rate (subroutine), 1138
node_component_hot_halo_variable_simple_post_evolve (subroutine), 1138
node_component_hot_halo_variable_simple_promote (subroutine), 1138
<table>
<thead>
<tr>
<th>Function/Method Name</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_component</td>
<td>hot_halo_very_simple_push_to_cooling_pipes</td>
<td>1138</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hot_halo_very_simple_rate_compute</td>
<td>1138</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hot_halo_very_simple_reset</td>
<td>1138</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hot_halo_very_simple_satellite_merger</td>
<td>1138</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hot_halo_very_simple_scale_set</td>
<td>1138</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hot_halo_very_simple_tree_initialize</td>
<td>1138</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalo_builder</td>
<td>1016</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalo_destroy</td>
<td>1016</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalo_dump</td>
<td>1017</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalo_initializer</td>
<td>1017</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalo_move</td>
<td>1017</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalo_output</td>
<td>1017</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalo_output_count</td>
<td>1017</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalo_output_names</td>
<td>1017</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalo_remove</td>
<td>1017</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalo_type</td>
<td>1017</td>
</tr>
<tr>
<td></td>
<td>function</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_builder</td>
<td>1017</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_count</td>
<td>1018</td>
</tr>
<tr>
<td></td>
<td>function</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_deserialize_rates</td>
<td>1018</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_deserialize_scales</td>
<td>1018</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_deserialize_values</td>
<td>1018</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_destroy</td>
<td>1018</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_dump</td>
<td>1018</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_dump_raw</td>
<td>1018</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_name_from_index</td>
<td>1018</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_ode_step_rates_init</td>
<td>1019</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_ode_step_scales_init</td>
<td>1019</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_output_count</td>
<td>1019</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_output_names</td>
<td>1019</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_read</td>
<td>1019</td>
</tr>
<tr>
<td></td>
<td>raw subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_serialize_rates</td>
<td>1019</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_serialize_scales</td>
<td>1019</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_serialize_values</td>
<td>1020</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalonull_type</td>
<td>1020</td>
</tr>
<tr>
<td></td>
<td>function</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalostandard_builder</td>
<td>1020</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalostandard_count</td>
<td>1020</td>
</tr>
<tr>
<td></td>
<td>function</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalostandard_deserialize_rates</td>
<td>1020</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalostandard_deserialize_scales</td>
<td>1020</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalostandard_deserialize_values</td>
<td>1020</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalostandard_destroy</td>
<td>1020</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalostandard_dump</td>
<td>1020</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalostandard_dump_raw</td>
<td>1021</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalostandard_initializer</td>
<td>1021</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalostandard_is_active</td>
<td>1021</td>
</tr>
<tr>
<td></td>
<td>function</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalostandard_name_from_index</td>
<td>1021</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalostandard_ode_step_rates_init</td>
<td>1021</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
<tr>
<td>node_component</td>
<td>hothalostandard_ode_step_scales_init</td>
<td>1021</td>
</tr>
<tr>
<td></td>
<td>subroutine</td>
<td></td>
</tr>
</tbody>
</table>
node_component_hothalostandard_output_count (subroutine), 1021
node_component_hothalostandard_output_names (subroutine), 1021
node_component_hothalostandard_read_raw (subroutine), 1021
node_component_hothalostandard_serialize_rates (subroutine), 1022
node_component_hothalostandard_serialize_scales (subroutine), 1022
node_component_hothalostandard_serialize_values (subroutine), 1022
node_component_hothalostandard_type (function), 1022
node_component_hothaloversiple_builder (subroutine), 1022
node_component_hothaloversiple_count (function), 1022
node_component_hothaloversiple_deserialize_rates (subroutine), 1022
node_component_hothaloversiple_deserialize_scales (subroutine), 1022
node_component_hothaloversiple_deserialize_values (subroutine), 1022
node_component_hothaloversiple_destroy (subroutine), 1023
node_component_hothaloversiple_dump (subroutine), 1023
node_component_hothaloversiple_dump_raw (subroutine), 1023
node_component_hothaloversiple_initializor (subroutine), 1023
node_component_hothaloversiple_is_active (function), 1023
node_component_hothaloversiple_name_from_index (subroutine), 1023
node_component_hothaloversiple_ode_step_rates_init (subroutine), 1023
node_component_hothaloversiple_ode_step_scales_init (subroutine), 1023
node_component_hothaloversiple_output_count (subroutine), 1024
node_component_hothaloversiple_output_names (subroutine), 1024
node_component_hothaloversiple_serialize_rates (subroutine), 1024
node_component_hothaloversiple_serialize_scales (subroutine), 1024
node_component_hothaloversiple_serialize_values (subroutine), 1024
node_component_hothaloversiple_type (function), 1024
node_component_indices_builder (subroutine), 1024
node_component_indices_destroy (subroutine), 1024
node_component_indices_dump (subroutine), 1025
node_component_indices_initializor (subroutine), 1025
node_component_indices_move (subroutine), 1025
node_component_indices_output (subroutine), 1025
node_component_indices_output_count (subroutine), 1025
node_component_indices_output_names (subroutine), 1025
node_component_indices_remove (subroutine), 1025
node_component_indices_standard (module), 1139
node_component_indices_standard_merger_tree_init (subroutine), 1139
node_component_indices_type (function), 1025
node_component_indicesnull_builder (subroutine), 1025
node_component_indicesnull_count (function), 1026
node_component_indicesnull_deserialize_rates (subroutine), 1026
node_component_indicesnull_deserialize_scales (subroutine), 1026
node_component_indicesnull_deserialize_values (subroutine), 1026
node_component_indicesnull_destroy (subroutine), 1026
node_component_indicesnull_dump (subroutine), 1026
node_component_indicesnull_dump_raw (subroutine), 1026
node_component_indicesnull_initializor (subroutine), 1026
node_component_indicesnull_is_active (function), 1027
CODE INDEX

node_component_indicesnull_name_from_index (subroutine), 1027
node_component_indicesnull_ode_step_rates_init (subroutine), 1027
node_component_indicesnull_ode_step_scales_init (subroutine), 1027
node_component_indicesnull_output_count (subroutine), 1027
node_component_indicesnull_output_names (subroutine), 1027
node_component_indicesnull_read_raw (subroutine), 1027
node_component_indicesnull_serialize_rates (subroutine), 1027
node_component_indicesnull_serialize_scales (subroutine), 1027
node_component_indicesnull_serialize_values (subroutine), 1027
node_component_indicesnull_type (function), 1028
node_component_indicesstandard_builder (subroutine), 1028
node_component_indicesstandard_count (function), 1028
node_component_indicesstandard_deserialize_rates (subroutine), 1028
node_component_indicesstandard_deserialize_scales (subroutine), 1028
node_component_indicesstandard_deserialize_values (subroutine), 1028
node_component_indicesstandard_destroy (subroutine), 1028
node_component_indicesstandard_dump (subroutine), 1028
node_component_indicesstandard_dump_raw (subroutine), 1029
node_component_indicesstandard_initializor (subroutine), 1029
node_component_indicesstandard_is_active (function), 1029
node_component_indicesstandard_name_from_index (subroutine), 1029
node_component_indicesstandard_ode_step_rates_init (subroutine), 1029
node_component_indicesstandard_ode_step_scales_init (subroutine), 1029
node_component_indicesstandard_output_count (subroutine), 1029
node_component_indicesstandard_output_names (subroutine), 1029
node_component_indicesstandard_read_raw (subroutine), 1029
node_component_indicesstandard_serialize_rates (subroutine), 1030
node_component_indicesstandard_serialize_scales (subroutine), 1030
node_component_indicesstandard_serialize_values (subroutine), 1030
node_component_indicesstandard_type (function), 1030
node_component_inter_output_standard (module), 1139
node_component_inter_output_standard_rate_compute (subroutine), 1139
node_component_inter_output_standard_rate_compute (subroutine), 1139
node_component_inter_output_standard_reset (subroutine), 1139
node_component_inter_output_standard_satellite_merger (subroutine), 1139
node_component_interoutput_builder (subroutine), 1030
node_component_interoutput_destroy (subroutine), 1030
node_component_interoutput_dump (subroutine), 1030
node_component_interoutput_initializor (subroutine), 1030
node_component_interoutput_move (subroutine), 1031
node_component_interoutput_output (subroutine), 1031
node_component_interoutput_output_count (subroutine), 1031
node_component_interoutput_output_names (subroutine), 1031
node_component_interoutput_remove (subroutine), 1031
node_component_interoutput_type (function), 1031
node_component_interoutputnull_builder (subroutine), 1031
node_component_interoutputnull_count (function), 1031
node_component_interoutputnull_deserialize_rates (subroutine), 1031
node_component_interoutputnull_deserialize_scales (subroutine), 1032
node_component_interoutputnull_deserialize_values (subroutine), 1032
node_component_interoutputnull_destroy (subroutine), 1032
node_component_interoutputnull_dump (subroutine), 1032
node_component_interoutputnull_dump_raw (subroutine), 1032
node_component_interoutputnull_initializor (subroutine), 1032
node_component_interoutputnull_is_active (function), 1032
node_component_interoutputnull_name_from_index (subroutine), 1032
node_component_interoutputnull_ode_step_rates_init (subroutine), 1032
node_component_interoutputnull_ode_step_scales_init (subroutine), 1033
node_component_interoutputnull_output_count (subroutine), 1033
node_component_interoutputnull_output_names (subroutine), 1033
node_component_interoutputnull_read_raw (subroutine), 1033
node_component_interoutputnull_serialize_rates (subroutine), 1033
node_component_interoutputnull_serialize_scales (subroutine), 1033
node_component_interoutputnull_serialize_values (subroutine), 1033
node_component_interoutputnull_type (function), 1033
node_component_interoutputstandard_builder (subroutine), 1033
node_component_interoutputstandard_count (function), 1034
node_component_interoutputstandard_deserialize_rates (subroutine), 1034
node_component_interoutputstandard_deserialize_scales (subroutine), 1034
node_component_interoutputstandard_deserialize_values (subroutine), 1034
node_component_interoutputstandard_destroy (subroutine), 1034
node_component_interoutputstandard_dump (subroutine), 1034
node_component_interoutputstandard_dump_raw (subroutine), 1034
node_component_interoutputstandard_initializor (subroutine), 1034
node_component_interoutputstandard_name_from_index (subroutine), 1035
node_component_interoutputstandard_ode_step_rates_init (subroutine), 1035
node_component_interoutputstandard_ode_step_scales_init (subroutine), 1035
node_component_interoutputstandard_output_count (subroutine), 1035
node_component_interoutputstandard_read_raw (subroutine), 1035
node_component_interoutputstandard_serialize_rates (subroutine), 1035
node_component_interoutputstandard_serialize_scales (subroutine), 1035
node_component_interoutputstandard_serialize_values (subroutine), 1036
node_component_interoutputstandard_type (function), 1036
node_component_merging_statistics_recent (module), 1140
node_component_merging_statistics_recent_initialize (subroutine), 1140
node_component_merging_statistics_recent_matches (function), 1140
node_component_merging_statistics_recent_merge_tree_init (subroutine), 1140
node_component_merging_statistics_recent_node_merge (subroutine), 1140
node_component_merging_statistics_recent_node_promotion (subroutine), 1140
node_component_merging_statistics_recent_output (subroutine), 1140
node_component_merging_statistics_recent_output_count (subroutine), 1141
node_component_merging_statistics_recent_output_names (subroutine), 1141
node_component_merging_statistics_standard (module), 1141
node_component_merging_statistics_standard_initialize (subroutine), 1141
node_component_merging_statistics_standard_merge_tree_init (subroutine), 1141
node_component_merging_statistics_standard_node_merge (subroutine), 1141
node_component_merging_statistics_standard_node_promotion (subroutine), 1141
node_component_merging_statistics_standard_satellite Merger (subroutine), 1142
node_component_mergingstatistics_builder (subroutine), 1036
node_component_mergingstatistics_destroy (subroutine), 1036
node_component_mergingstatistics_dump (subroutine), 1036
node_component_mergingstatistics_initializer (subroutine), 1036
node_component_mergingstatistics_move (subroutine), 1036
node_component_mergingstatistics_output (subroutine), 1036
node_component_mergingstatistics_output_count (subroutine), 1037
node_component_mergingstatistics_output_names (subroutine), 1037
node_component_mergingstatistics_remove (subroutine), 1037
node_component_mergingstatistics_type (function), 1037
node_component_mergingstatisticsnull_builder (subroutine), 1037
node_component_mergingstatisticsnull_count (function), 1037
node_component_mergingstatisticsnull_deserialize_rates (subroutine), 1037
node_component_mergingstatisticsnull_deserialize_scales (subroutine), 1037
node_component_mergingstatisticsnull_deserialize_values (subroutine), 1037
node_component_mergingstatisticsnull_destroy (subroutine), 1038
node_component_mergingstatisticsnull_dump (subroutine), 1038
node_component_mergingstatisticsnull_dump_raw (subroutine), 1038
node_component_mergingstatisticsnull_initializor (subroutine), 1038
node_component_mergingstatisticsnull_is_active (function), 1038
node_component_mergingstatisticsnull_name_from_index (subroutine), 1038
node_component_mergingstatisticsnull_ode_step_rates_init (subroutine), 1038
node_component_mergingstatisticsnull_ode_step_scales_init (subroutine), 1038
node_component_mergingstatisticsnull_output_count (subroutine), 1039
node_component_mergingstatisticsnull_output_names (subroutine), 1039
node_component_mergingstatisticsnull_read_raw (subroutine), 1039
node_component_mergingstatisticsnull_serialize_rates (subroutine), 1039
node_component_mergingstatisticsnull_serialize_scales (subroutine), 1039
node_component_mergingstatisticsnull_serialize_values (subroutine), 1039
node_component_mergingstatisticsnull_type (function), 1039
node_component_mergingstatisticsrecent_builder (subroutine), 1039
node_component_mergingstatisticsrecent_count (function), 1039
node_component_mergingstatisticsrecent_deserialize_rates (subroutine), 1040
node_component_mergingstatisticsrecent_deserialize_scales (subroutine), 1040
node_component_mergingstatisticsrecent_deserialize_values (subroutine), 1040
node_component_mergingstatisticsrecent_destroy (subroutine), 1040
node_component_mergingstatisticsrecent_dump (subroutine), 1040
node_component_mergingstatisticsrecent_dump_raw (subroutine), 1040
node_component_mergingstatisticsrecent_initializor (subroutine), 1040
node_component_mergingstatisticsrecent_is_active (function), 1040
node_component_mergingstatisticsrecent_name_from_index (subroutine), 1040
node_component_mergingstatisticsrecent_ode_step_rates_init (subroutine), 1041
node_component_mergingstatisticsrecent_ode_step_scales_init (subroutine), 1041
node_component_mergingstatisticsrecent_output_count (subroutine), 1041
node_component_mergingstatisticsrecent_output_names (subroutine), 1041
node_component_mergingstatisticsrecent_read_raw (subroutine), 1041
node_component_mergingstatisticsrecent_serialize_rates (subroutine), 1041
node_component_mergingstatisticsrecent_serialize_scales (subroutine), 1041
node_component_mergingstatisticsrecent_serialize_values (subroutine), 1041
node_component_mergingstatisticsrecent_type (function), 1041
node_component_mergingstatisticsstandard_builder (subroutine), 1042
node_component_mergingstatisticsstandard_count (function), 1042
node_component_mergingstatisticsstandard_deserialize_rates (subroutine), 1042
node_component_mergingstatisticsstandard_deserialize_scales (subroutine), 1042
node_component_mergingstatisticsstandard_deserialize_values (subroutine), 1042
node_component_mergingstatisticsstandard_destroy (subroutine), 1042
node_component_mergingstatisticsstandard_dump (subroutine), 1042
node_component_mergingstatisticsstandard_dump_raw (subroutine), 1042
node_component_mergingstatisticsstandard_initializor (subroutine), 1043
node_component_mergingstatisticsstandard_is_active (function), 1043
node_component_mergingstatisticsstandard_name_from_index (subroutine), 1043
node_component_mergingstatisticsstandard_ode_step_rates_init (subroutine), 1043
node_component_mergingstatisticsstandard_ode_step_scales_init (subroutine), 1043
node_component_mergingstatisticsstandard_output_count (subroutine), 1043
node_component_mergingstatisticsstandard_output_names (subroutine), 1043
node_component_mergingstatisticsstandard_read_raw (subroutine), 1043
node_component_mergingstatisticsstandard_serialize_rates (subroutine), 1044
node_component_mergingstatisticsstandard_serialize_scales (subroutine), 1044
node_component_mergingstatisticsstandard_serialize_values (subroutine), 1044
node_component_mergingstatisticsstandard_type (function), 1044
node_component_name_from_index (subroutine), 1044
node_component_null_double0_inout (function), 1044
node_component_null_void0_inout (subroutine), 1044
node_component_ode_step_initialize_null (subroutine), 1044
node_component_output_count_null (subroutine), 1044
node_component_output_names_null (subroutine), 1044
node_component_position_builder (subroutine), 1045
node_component_position_destroy (subroutine), 1045
node_component_position_dump (subroutine), 1045
node_component_position_initializor (subroutine), 1045
node_component_position_move (subroutine), 1045
node_component_position_output (subroutine), 1045
node_component_position_output_count (subroutine), 1045
node_component_position_output_names (subroutine), 1045
node_component_position_preset (module), 1142
node_component_position_preset_node_promotion (subroutine), 1142
node_component_position_remove (subroutine), 1046
node_component_position_type (function), 1046
node_component_positionnull_builder (subroutine), 1046
node_component_positionnull_count (function), 1046
node_component_positionnull_deserialize_rates (subroutine), 1046
node_component_positionnull_deserialize_scales (subroutine), 1046
node_component_positionnull_deserialize_values (subroutine), 1046
node_component_positionnull_destroy (subroutine), 1046
node_component_positionnull_dump (subroutine), 1047
node_component_positionnull_dump_raw (subroutine), 1047
node_component_positionnull_initializor (subroutine), 1047
node_component_positionnull_is_active (function), 1047
node_component_positionnull_name_from_index (subroutine), 1047
node_component_positionnull_ode_step_rates_init (subroutine), 1047
node_component_positionnull_ode_step_scales_init (subroutine), 1047
node_component_positionnull_output_count (subroutine), 1047
node_component_positionnull_output_names (subroutine), 1047
node_component_positionnull_read_raw (subroutine), 1048
node_component_positionnull_serialize_rates (subroutine), 1048
node_component_positionnull_serialize_scales (subroutine), 1048
node_component_positionnull_serialize_values (subroutine), 1048
node_component_positionnull_type (function), 1048
node_component_positionpreset_builder (subroutine), 1048
node_component_positionpreset_count (function), 1048
node_component_positionpreset_deserialize_rates (subroutine), 1048
node_component_positionpreset_deserialize_scales (subroutine), 1048
node_component_positionpreset_deserialize_values (subroutine), 1049
node_component_positionpreset_destroy (subroutine), 1049
node_component_positionpreset_dump (subroutine), 1049
node_component_positionpreset_dump_raw (subroutine), 1049
node_component_positionpreset_initializor (subroutine), 1049
node_component_positionpreset_is_active (function), 1049
node_component_positionpreset_name_from_index (subroutine), 1049
node_component_positionpreset_ode_step_rates_init (subroutine), 1049
node_component_positionpreset_ode_step_scales_init (subroutine), 1049
node_component_positionpreset_output_count (subroutine), 1050
node_component_positionpreset_output_names (subroutine), 1050
node_component_positionpreset_read_raw (subroutine), 1050
node_component_positionpreset_serialize_rates (subroutine), 1050
node_component_positionpreset_serialize_scales (subroutine), 1050
node_component_positionpreset_serialize_values (subroutine), 1050
node_component_positionpreset_type (function), 1050
node_component_potential_null (function), 1050
node_component_rotation_curve_gradient_null (function), 1050
node_component_rotation_curve_null (function), 1051
node_component_satellite_builder (subroutine), 1051
node_component_satellite_destroy (subroutine), 1051
node_component_satellite_dump (subroutine), 1051
node_component_satellite_initializor (subroutine), 1051
node_component_satellite_move (subroutine), 1051
node_component_satellite_output (subroutine), 1051
node_component_satellite_output_count (subroutine), 1051
node_component_satellite_output_names (subroutine), 1051
node_component_satellite_preset (module), 1142
node_component_satellite_preset_promote (subroutine), 1142
node_component_satellite_remove (subroutine), 1052
node_component_satellite_standard (module), 1142
node_component_satellite_standard_create (subroutine), 1143
node_component_satellite_standard_halo_formation_task (subroutine), 1143
node_component_satellite_standard_initialize (subroutine), 1143
node_component_satellite_standard_rate_compute (subroutine), 1143
node_component_satellite_standard_scale_set (subroutine), 1143
node_component_satellite_standard_tree_initialize (subroutine), 1143
node_component_satellite_standard_virial_orbit (function), 1143
node_component_satellite_standard_virial_orbit_set (subroutine), 1144
node_component_satellite_type (function), 1052
node_component_satellite_very_simple (module), 1144
node_component_satellite_very_simple_create (subroutine), 1144
node_component_satellite_very_simple_halosformation_task (subroutine), 1144
node_component_satellite_very_simple_initialize (subroutine), 1144
node_component_satellite_very_simple_tree_initialize (subroutine), 1144
node_component_satellitenull_builder (subroutine), 1052
node_component_satellitenull_count (function), 1052
node_component_satellitenull_deserialize_rates (subroutine), 1052
node_component_satellitenull_deserialize_scales (subroutine), 1052
node_component_satellitenull_deserialize_values (subroutine), 1052
node_component_satellitenull_destroy (subroutine), 1052
node_component_satellitenull_dump (subroutine), 1052
node_component_satellitenull_dump_raw (subroutine), 1053
node_component_satellitenull_initializor (subroutine), 1053
node_component_satellitenull_is_active (function), 1053
node_component_satellitenull_name_from_index (subroutine), 1053
node_component_satellitenull_ode_step_rates_init (subroutine), 1053
node_component_satellitenull_ode_step_scales_init (subroutine), 1053
node_component_satellitenull_output_count (subroutine), 1053
node_component_satellitenull_output_names (subroutine), 1053
node_component_satellitenull_read_raw (subroutine), 1053
node_component_satellitenull_serialize_rates (subroutine), 1054
node_component_satellitenull_serialize_scales (subroutine), 1054
node_component_satellitenull_serialize_values (subroutine), 1054
node_component_satellitenull_type (function), 1054
node_component_satellitepreset_builder (subroutine), 1054
node_component_satellitepreset_count (function), 1054
node_component_satellitepreset_deserialize_rates (subroutine), 1054
node_component_satellitepreset_deserialize_scales (subroutine), 1054
node_component_satellitepreset_deserialize_values (subroutine), 1054
node_component_satellitepreset_destroy (subroutine), 1055
node_component_satellitepreset_dump (subroutine), 1055
node_component_satellitepreset_dump_raw (subroutine), 1055
node_component_satellitepreset_initializor (subroutine), 1055
node_component_satellitepreset_is_active (function), 1055
node_component_satellitepreset_name_from_index (subroutine), 1055
node_component_satellitepreset_ode_step_rates_init (subroutine), 1055
node_component_satellitepreset_ode_step_scales_init (subroutine), 1055
node_component_satellitepreset_output_count (subroutine), 1055
node_component_satellitepreset_output_names (subroutine), 1056
node_component_satellitepreset_read_raw (subroutine), 1056
node_component_satellitepreset_serialize_rates (subroutine), 1056
CODE INDEX

node_component_satellitepreset_serialize_scales (subroutine), 1056
node_component_satellitepreset_serialize_values (subroutine), 1056
node_component_satellitepreset_type (function), 1056
node_component_satellitepreset_builder (subroutine), 1056
node_component_satellitepreset_count (function), 1056
node_component_satellitepreset_deserialize_rates (subroutine), 1056
node_component_satellitepreset_deserialize_scales (subroutine), 1056
node_component_satellitepreset_deserialize_values (subroutine), 1056
node_component_satellitepreset_destroy (subroutine), 1057
node_component_satellitepreset_dump (subroutine), 1057
node_component_satellitepreset_dump_raw (subroutine), 1057
node_component_satellitepreset_initializor (subroutine), 1057
node_component_satellitepreset_is_active (function), 1057
node_component_satellitepreset_name_from_index (subroutine), 1057
node_component_satellitepreset_ode_step_rates_init (subroutine), 1058
node_component_satellitepreset_ode_step_scales_init (subroutine), 1058
node_component_satellitepreset_output_count (subroutine), 1058
node_component_satellitepreset_output_names (subroutine), 1058
node_component_satellitepreset_read_raw (subroutine), 1058
node_component_satellitepreset_serialize_rates (subroutine), 1058
node_component_satellitepreset_serialize_scales (subroutine), 1058
node_component_satellitepreset_serialize_values (subroutine), 1058
node_component_satellitepreset_type (function), 1058
node_component_satelliteverysimple_builder (subroutine), 1059
node_component_satelliteverysimple_count (function), 1059
node_component_satelliteverysimple_deserialize_rates (subroutine), 1059
node_component_satelliteverysimple_deserialize_scales (subroutine), 1059
node_component_satelliteverysimple_deserialize_values (subroutine), 1059
node_component_satelliteverysimple_destroy (subroutine), 1059
node_component_satelliteverysimple_dump (subroutine), 1059
node_component_satelliteverysimple_dump_raw (subroutine), 1059
node_component_satelliteverysimple_initializor (subroutine), 1059
node_component_satelliteverysimple_is_active (function), 1060
node_component_satelliteverysimple_name_from_index (subroutine), 1060
node_component_satelliteverysimple_ode_step_rates_init (subroutine), 1060
node_component_satelliteverysimple_ode_step_scales_init (subroutine), 1060
node_component_satelliteverysimple_output_count (subroutine), 1060
node_component_satelliteverysimple_output_names (subroutine), 1060
node_component_satelliteverysimple_read_raw (subroutine), 1060
node_component_satelliteverysimple_serialize_rates (subroutine), 1060
node_component_satelliteverysimple_serialize_scales (subroutine), 1061
node_component_satelliteverysimple_serialize_values (subroutine), 1061
node_component_satelliteverysimple_type (function), 1061
node_component_serialize_count_zero (function), 1061
node_component_serialize_null (subroutine), 1061
node_component_spheroid_builder (subroutine), 1061
node_component_spheroid_destroy (subroutine), 1061
node_component_spheroid_dump (subroutine), 1061
node_component_spheroid_initializor (subroutine), 1061
node_component_spheroid_move (subroutine), 1062
node_component_spheroid_output (subroutine), 1062
node_component_spheroid_output_count (subroutine), 1062
node_component_spheroid_output_names (subroutine), 1062
node_component_spheroid_remove (subroutine), 1062
node_component_spheroid_standard (module), 1144
node_component_spheroid_standard_data (module), 1147
node_component_spheroid_standard_energy_gas_input_rate (subroutine), 1145
node_component_spheroid_standard_initialize (subroutine), 1145
node_component_spheroid_standard_initializor (subroutine), 1145
node_component_spheroid_standard_mass_gas_sink_rate (subroutine), 1145
node_component_spheroid_standard_post_evolve (subroutine), 1145
node_component_spheroid_standard_pre_evolve (subroutine), 1146
node_component_spheroid_standard_radius_solve (function), 1146
node_component_spheroid_standard_radius_solve_set (subroutine), 1146
node_component_spheroid_standard_radius_solver (subroutine), 1146
node_component_spheroid_standard_radius_solver_plausibility (subroutine), 1146
node_component_spheroid_standard_rate_compute (subroutine), 1146
node_component_spheroid_standard_satellite_merging (subroutine), 1146
node_component_spheroid_standard_scale_set (subroutine), 1146
node_component_spheroid_standard_mass_gas_sink_rate (subroutine), 1146
node_component_spheroid_standard_post_evolve (subroutine), 1146
node_component_spheroid_standard_pre_evolve (subroutine), 1147
node_component_spheroid_standard_star_formation_history_extend (subroutine), 1147
node_component_spheroid_standard_star_formation_history_output (subroutine), 1147
node_component_spheroid_standard_star_formation_history_rate (subroutine), 1147
node_component_spheroid_standard_star_formation_rate (function), 1147
node_component_spheroid_standard_velocity_solve (function), 1147
node_component_spheroid_standard_velocity_solve_set (subroutine), 1147
node_component_spheroid_type (function), 1062
node_component_spheroidnull_builder (subroutine), 1062
node_component_spheroidnull_count (function), 1062
node_component_spheroidnull_deserialize_rates (subroutine), 1063
node_component_spheroidnull_deserialize_scales (subroutine), 1063
node_component_spheroidnull_deserialize_values (subroutine), 1063
node_component_spheroidnull_destroy (subroutine), 1063
node_component_spheroidnull_dump (subroutine), 1063
node_component_spheroidnull_dump_raw (subroutine), 1063
node_component_spheroidnull_initializor (subroutine), 1063
node_component_spheroidnull_is_active (function), 1063
node_component_spheroidnull_name_from_index (subroutine), 1063
node_component_spheroidnull_ode_step_rates_init (subroutine), 1064
node_component_spheroidnull_ode_step_scales_init (subroutine), 1064
node_component_spheroidnull_output_count (subroutine), 1064
node_component_spheroidnull_output_names (subroutine), 1064
node_component_spheroidnull_read_raw (subroutine), 1064
node_component_spheroidnull_serialize_rates (subroutine), 1064
node_component_spheroidnull_serialize_scales (subroutine), 1064
node_component_spheroidnull_serialize_values (subroutine), 1064
node_component_spheroidnull_type (function), 1064
node_component_spheroidstandard_builder (subroutine), 1064
node_component_spheroidstandard_count (function), 1065
CODE INDEX

node_component_spheroidstandard_deserialize_rates (subroutine), 1065
node_component_spheroidstandard_deserialize_scales (subroutine), 1065
node_component_spheroidstandard_deserialize_values (subroutine), 1065
node_component_spheroidstandard_destroy (subroutine), 1065
node_component_spheroidstandard_dump (subroutine), 1065
node_component_spheroidstandard_dump_raw (subroutine), 1065
node_component_spheroidstandard_initializor (subroutine), 1065
node_component_spheroidstandard_is_active (function), 1066
node_component_spheroidstandard_name_from_index (subroutine), 1066
node_component_spheroidstandard_ode_step_rates_init (subroutine), 1066
node_component_spheroidstandard_ode_step_scales_init (subroutine), 1066
node_component_spheroidstandard_output_count (subroutine), 1066
node_component_spheroidstandard_output_names (subroutine), 1066
node_component_spheroidstandard_read_raw (subroutine), 1066
node_component_spheroidstandard_serialize_rates (subroutine), 1066
node_component_spheroidstandard_serialize_scales (subroutine), 1067
node_component_spheroidstandard_serialize_values (subroutine), 1067
node_component_spheroidstandard_type (function), 1067
node_component_spin_builder (subroutine), 1067
node_component_spin_destroy (subroutine), 1067
node_component-spin_dump (subroutine), 1067
node_component-spin_initializor (subroutine), 1067
node_component-spin_move (subroutine), 1067
node_component-spin_output (subroutine), 1067
node_component-spin_output_count (subroutine), 1068
node_component-spin_output_names (subroutine), 1068
node_component-spin_preset (module), 1148
node_component-spin_preset_initialize (subroutine), 1148
node_component-spin_preset_promote (subroutine), 1148
node_component-spin_preset_rate_compute (subroutine), 1148
node_component-spin_preset_scale_set (subroutine), 1148
node_component-spin_random (module), 1148
node_component-spin_random_initialize (subroutine), 1148
node_component-spin_random_initialize_spins (subroutine), 1149
node_component-spin_random_promote (subroutine), 1149
node_component-spin_remove (subroutine), 1068
node_component-spin_type (function), 1068
node_component_spinnull_builder (subroutine), 1068
node_component_spinnull_count (function), 1068
node_component_spinnull_deserialize_rates (subroutine), 1068
node_component_spinnull_deserialize_scales (subroutine), 1068
node_component_spinnull_deserialize_values (subroutine), 1068
node_component_spinnull_destroy (subroutine), 1069
node_component_spinnull_dump (subroutine), 1069
node_component_spinnull_dump_raw (subroutine), 1069
node_component_spinnull_initializor (subroutine), 1069
node_component_spinnull_is_active (function), 1069
node_component_spinnull_name_from_index (subroutine), 1069
node_component_spinnull_ode_step_rates_init (subroutine), 1069
node_component_spinnull_ode_step_scales_init (subroutine), 1069
node_component_spinnull_output_count (subroutine), 1069
node_component_spinnull_output_names (subroutine), 1070
node_component_spinnull_read_raw (subroutine), 1070
node_component_spinnull_serialize_rates (subroutine), 1070
node_component_spinnull_serialize_scales (subroutine), 1070
node_component_spinnull_serialize_values (subroutine), 1070
node_component_spinnull_type (function), 1070
node_component_spinpreset_builder (subroutine), 1070
node_component_spinpreset_count (function), 1070
node_component_spinpreset_deserialize_rates (subroutine), 1070
node_component_spinpreset_deserialize_scales (subroutine), 1071
node_component_spinpreset_deserialize_values (subroutine), 1071
node_component_spinpreset_destroy (subroutine), 1071
node_component_spinpreset_dump (subroutine), 1071
node_component_spinpreset_dump_raw (subroutine), 1071
node_component_spinpreset_initializor (subroutine), 1071
node_component_spinpreset_is_active (function), 1071
node_component_spinpreset_name_from_index (subroutine), 1071
node_component_spinpreset_ode_step_rates_init (subroutine), 1071
node_component_spinpreset_ode_step_scales_init (subroutine), 1072
node_component_spinpreset_output_count (subroutine), 1072
node_component_spinpreset_output_names (subroutine), 1072
node_component_spinpreset_read_raw (subroutine), 1072
node_component_spinpreset_serialize_rates (subroutine), 1072
node_component_spinpreset_serialize_scales (subroutine), 1072
node_component_spinpreset_serialize_values (subroutine), 1072
node_component_spinpreset_type (function), 1072
node_component_spinrandom_builder (subroutine), 1072
node_component_spinrandom_count (function), 1073
node_component_spinrandom_deserialize_rates (subroutine), 1073
node_component_spinrandom_deserialize_scales (subroutine), 1073
node_component_spinrandom_deserialize_values (subroutine), 1073
node_component_spinrandom_destroy (subroutine), 1073
node_component_spinrandom_dump (subroutine), 1073
node_component_spinrandom_dump_raw (subroutine), 1073
node_component_spinrandom_initializor (subroutine), 1073
node_component_spinrandom_is_active (function), 1073
node_component_spinrandom_name_from_index (subroutine), 1074
node_component_spinrandom_ode_step_rates_init (subroutine), 1074
node_component_spinrandom_ode_step_scales_init (subroutine), 1074
node_component_spinrandom_output_count (subroutine), 1074
node_component_spinrandom_output_names (subroutine), 1074
node_component_spinrandom_read_raw (subroutine), 1074
node_component_spinrandom_serialize_rates (subroutine), 1074
node_component_spinrandom_serialize_scales (subroutine), 1074
node_component_spinrandom_serialize_values (subroutine), 1074
node_component_spinrandom_type (function), 1075
node_component_surface_density_null (function), 1075
node_definition_index (function), 837
node_dump (subroutine), 1075
node_dump_raw (subroutine), 1075
node_location (function), 843
node_lookup (function), 837
node_output (subroutine), 1075
node_output_count (subroutine), 1075
node_output_names (subroutine), 1075
node_promotion_index_shift (subroutine), 718
node_promotion_index_shifts (module), 718
node_property_name_from_index (function), 1075
node_read_raw (subroutine), 1075
node_subhalo_promotion (function), 719
node_subhalo_promotions (module), 719
nodecomponent (type), 1076
nodecomponentbasic (type), 1076
nodecomponentbasicnonevolving (type), 1076
nodecomponentbasicnull (type), 1076
nodecomponentbasicstandard (type), 1076
nodecomponentblackhole (type), 1076
nodecomponentblackholenull (type), 1076
nodecomponentblackholesimple (type), 1076
nodecomponentblackholestandard (type), 1076
nodecomponentdarkmatterprofile (type), 1076
nodecomponentdarkmatterprofilenull (type), 1077
nodecomponentdarkmatterprofilescale (type), 1077
nodecomponentdarkmatterprofilesscaleshape (type), 1077
nodecomponentdisk (type), 1077
nodecomponentdiskexponential (type), 1077
nodecomponentdisknull (type), 1077
nodecomponentdiskverysimple (type), 1077
nodecomponentformationtime (type), 1077
nodecomponentformationtimecole2000 (type), 1077
nodecomponentformationtimenull (type), 1077
nodecomponenthothalo (type), 1078
nodecomponenthothalonull (type), 1078
nodecomponenthothalostandard (type), 1078
nodecomponenthothaloverysimple (type), 1078
nodecomponentindices (type), 1078
nodecomponentindicesnull (type), 1078
nodecomponentindicesstandard (type), 1078
nodecomponentinteroutput (type), 1078
nodecomponentinteroutputnull (type), 1078
nodecomponentinteroutputstandard (type), 1078
nodecomponentmergingstatistics (type), 1079
nodecomponentmergingstatisticsnull (type), 1079
nodecomponentmergingstatisticsrecent (type), 1079
nodecomponentmergingstatisticsstandard (type), 1079
nodecomponentposition (type), 1079
nodecomponentpositionnull (type), 1079
nodecomponentpositionpreset (type), 1079
nodecomponentsatellite (type), 1079
nodecomponentsatellitenvnull (type), 1079
nodecomponentsatellitepreset (type), 1080
nodecomponentsatellitesatellite (type), 1080
nodecomponentsatelliteverysimple (type), 1080
nodecomponentspheroid (type), 1080
nodecomponentspheroidnull (type), 1080
nodecomponentspheroidstandard (type), 1080
nodecompontspin (type), 1080
nodecomponentspinpnull (type), 1080
nodecomponentspinpreset (type), 1080
nodecomponentspinrandom (type), 1080
nodedata (type), 843
nodedataabundanceescalarevolvable (type), 1081
nodedatachemicalabundanceescalarevolvable (type), 1081
nodedatadouble1d (type), 1081
nodedatadouble1devolvable (type), 1081
nodedatatodoublescalar (type), 1081
nodedatatodoublescalarevolvable (type), 1081
nodedatahistoryscalar (type), 1081
nodedatahistoryscalarevolvable (type), 1081
nodedataintegerid (type), 1081
nodedataintegerscalar (type), 1082
nodedatakeplerorbit (type), 1082
nodedatatime (type), 1082
nodeevent (type), 1082
number_density_critical_over_number_density_hydrogen (subroutine), 654
numerical.comparison.F90 (file), 863
numerical.constants.astronomical.F90 (file), 864
numerical.constants.atomic.F90 (file), 866
numerical.constants.math.F90 (file), 866
numerical.constants.physical.F90 (file), 868
numerical.constants.prefixes.F90 (file), 870
numerical.constants.units.F90 (file), 870
numerical.fft3.F90 (file), 871
numerical.integration.F90 (file), 871
numerical.interpolation.2D.irregular.F90 (file), 872
numerical.interpolation.F90 (file), 873
numerical.meshes.F90 (file), 875
numerical.ODE_solver.error_codes.F90 (file), 863
numerical.ODE_solver.F90 (file), 858
numerical.ODE_solver.ODEIV2.F90 (file), 859
numerical.ODE_solver.ODEIV2.wrapper.F90 (file), 859
numerical.random.F90 (file), 876
numerical.random.gaussian.F90 (file), 877
numerical.random.quasi.F90 (file), 877
<table>
<thead>
<tr>
<th>Module/Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>numerical.ranges.F90</code></td>
<td>878</td>
</tr>
<tr>
<td><code>numerical.root_finder.F90</code></td>
<td>879</td>
</tr>
<tr>
<td><code>numerical.search.F90</code></td>
<td>881</td>
</tr>
<tr>
<td><code>numerical.sort.F90</code></td>
<td>882</td>
</tr>
<tr>
<td><code>numerical_comparison</code></td>
<td>863</td>
</tr>
<tr>
<td><code>numerical_constants_astronomical</code></td>
<td>864</td>
</tr>
<tr>
<td><code>numerical_constants_atomic</code></td>
<td>866</td>
</tr>
<tr>
<td><code>numerical_constants_math</code></td>
<td>866</td>
</tr>
<tr>
<td><code>numerical_constants_physical</code></td>
<td>868</td>
</tr>
<tr>
<td><code>numerical_constants_prefixes</code></td>
<td>870</td>
</tr>
<tr>
<td><code>numerical_constants_units</code></td>
<td>870</td>
</tr>
<tr>
<td><code>numerical_integration</code></td>
<td>871</td>
</tr>
<tr>
<td><code>numerical_interpolation</code></td>
<td>873</td>
</tr>
<tr>
<td><code>numerical_interpolation_2d_irregular</code></td>
<td>872</td>
</tr>
<tr>
<td><code>numerical_ranges</code></td>
<td>878</td>
</tr>
<tr>
<td><code>objects.abundances.F90</code></td>
<td>883</td>
</tr>
<tr>
<td><code>objects.chemical_abundances.F90</code></td>
<td>889</td>
</tr>
<tr>
<td><code>objects.chemical_structure.F90</code></td>
<td>893</td>
</tr>
<tr>
<td><code>objects.coordinates.F90</code></td>
<td>894</td>
</tr>
<tr>
<td><code>objects.history.F90</code></td>
<td>898</td>
</tr>
<tr>
<td><code>objects.kepler_orbits.F90</code></td>
<td>902</td>
</tr>
<tr>
<td><code>objects.mass_distributions.F90</code></td>
<td>907</td>
</tr>
<tr>
<td><code>objects.merger_tree_data.F90</code></td>
<td>909</td>
</tr>
<tr>
<td><code>objects.merger_trees.dump.F90</code></td>
<td>913</td>
</tr>
<tr>
<td><code>objects.merger_trees.information_content.F90</code></td>
<td>914</td>
</tr>
<tr>
<td><code>objects.nodes.bindings.C.F90</code></td>
<td>1113</td>
</tr>
<tr>
<td><code>objects.nodes.components.basic.non_evolving.F90</code></td>
<td>1118</td>
</tr>
<tr>
<td><code>objects.nodes.components.basic.standard.F90</code></td>
<td>1119</td>
</tr>
<tr>
<td><code>objects.nodes.components.black_hole.simple.F90</code></td>
<td>1120</td>
</tr>
<tr>
<td><code>objects.nodes.components.black_hole.simple.structure.F90</code></td>
<td>1121</td>
</tr>
<tr>
<td><code>objects.nodes.components.black_hole.standard.F90</code></td>
<td>1122</td>
</tr>
<tr>
<td><code>objects.nodes.components.black_hole.standard.structure_tasks.F90</code></td>
<td>1124</td>
</tr>
<tr>
<td><code>objects.nodes.components.dark_matter_profile.scale.F90</code></td>
<td>1125</td>
</tr>
<tr>
<td><code>objects.nodes.components.dark_matter_profile.scale_shape.F90</code></td>
<td>1127</td>
</tr>
<tr>
<td><code>objects.nodes.components.disk.exponential.data.F90</code></td>
<td>1130</td>
</tr>
<tr>
<td><code>objects.nodes.components.disk.exponential.F90</code></td>
<td>1128</td>
</tr>
<tr>
<td><code>objects.nodes.components.disk.very_simple.F90</code></td>
<td>1131</td>
</tr>
<tr>
<td><code>objects.nodes.components.formation_times.Cole2000.F90</code></td>
<td>1133</td>
</tr>
<tr>
<td><code>objects.nodes.components.hot_halo.standard.F90</code></td>
<td>1133</td>
</tr>
<tr>
<td><code>objects.nodes.components.hot_halo.very_simple.F90</code></td>
<td>1137</td>
</tr>
<tr>
<td><code>objects.nodes.components.indices.standard.F90</code></td>
<td>1139</td>
</tr>
<tr>
<td><code>objects.nodes.components.interoutput.standard.F90</code></td>
<td>1139</td>
</tr>
<tr>
<td><code>objects.nodes.components.merging_statistics.recent.F90</code></td>
<td>1140</td>
</tr>
<tr>
<td><code>objects.nodes.components.merging_statistics.standard.F90</code></td>
<td>1141</td>
</tr>
<tr>
<td><code>objects.nodes.components.position.preset.F90</code></td>
<td>1142</td>
</tr>
<tr>
<td><code>objects.nodes.components.satellite.preset.F90</code></td>
<td>1142</td>
</tr>
<tr>
<td><code>objects.nodes.components.satellite.standard.F90</code></td>
<td>1142</td>
</tr>
<tr>
<td><code>objects.nodes.components.satellite.very_simple.F90</code></td>
<td>1144</td>
</tr>
<tr>
<td><code>objects.nodes.components.spheroid.standard.data.F90</code></td>
<td>1147</td>
</tr>
</tbody>
</table>
objects.nodes.components.spheroid.standard.F90 (file), 1144
objects.nodes.components.spin.preset.F90 (file), 1148
objects.nodes.components.spin.random.F90 (file), 1148
objects.nodes.F90 (file), 914
objects.radiation.F90 (file), 1149
objects.tables.F90 (file), 1152
ode_set_1 (function), 1284
ode_set_2 (function), 1284
ode_solve (subroutine), 858
ode_solver (module), 858
ode_solver_error_codes (module), 863
ode_solver_free (subroutine), 858
odeiv2_solve (subroutine), 859
odeiv2_solver (module), 859
omega_b (function), 685
omega_dark_energy (function), 676
omega_dark_energy_matter_dark_energy (function), 679
omega_dark_energy_matter_lambda (function), 682
omega_de (function), 685
omega_k (function), 685
omega_matter (function), 685
omega_matter_total (function), 676
omega_matter_total_matter_dark_energy (function), 679
omega_matter_total_matter_lambda (function), 683
omega_radiation (function), 686
op_assign_ch_vs (subroutine), 816
op_assign_vs_ch (subroutine), 817
op_concat_ch_vs (function), 817
op_concat_vs_ch (function), 817
op_concat_vs_vs (function), 817
op_eq_ch_vs (function), 817
op_eq_vs_ch (function), 817
op_eq_vs_vs (function), 817
op_ge_ch_vs (function), 817
op_ge_vs_ch (function), 817
op_ge_vs_vs (function), 817
op_gt_ch_vs (function), 817
op_gt_vs_ch (function), 817
op_gt_vs_vs (function), 818
op_le_ch_vs (function), 818
op_le_vs_ch (function), 818
op_le_vs_vs (function), 818
op_lt_ch_vs (function), 818
op_lt_vs_ch (function), 818
op_lt_vs_vs (function), 818
op_ne_ch_vs (function), 818
op_ne_vs_ch (function), 818
op_ne_vs_vs (function), 818
operator (interface), 818
operator(*) (interface), 889
operator(/) (interface), 818, 1357
operator(/=) (interface), 819
operator(==) (interface), 819
Optimal_Sampling.Stellar_Mass_Function.F90 (file), 600
optimal_sampling_smf (program), 600
output_times_initialize (subroutine), 778
outputgroup (type), 759
perform_node_events (subroutine), 847
pericenter_solver (function), 1180
perturbation_dynamics_solver (subroutine), 1272
perturbation_integrand (function), 1274
perturbation_maximum_radius (function), 1274
perturbationnodes (function), 1272
piecewise_power_law_imf_normalize (subroutine), 1197
piecewise_power_law_imf_phi (interface), 1197
piecewise_power_law_imf_phi_array (function), 1197
piecewise_power_law_imf_phi_scalar (function), 1197
pointed_at_node (function), 846
positioncountlinked (function), 1082
positioncreatebyinterrupt (subroutine), 1082
positioncreatelinked (subroutine), 1082
positiondestroylinked (subroutine), 1082
positionget (function), 1083
positionnullbindingdouble1inout (function), 1083
positionnullbindinghistory0inout (function), 1083
positionnullbindinginteger0in (function), 1083
positionnullbindingratedouble1inout (subroutine), 1083
positionnullbindingratehistory0inout (subroutine), 1083
positionnullbindingrateinteger0in (subroutine), 1083
positionnullbindingsetdouble1inout (subroutine), 1083
positionnullbindingsethistory0inout (subroutine), 1083
positionnullbindingsetinteger0in (subroutine), 1083
positionposition (function), 1084
positionpositionhistory (function), 1084
positionpositionhistoryisgettable (function), 1084
positionpositionisgettable (function), 1084
positionpresetpositionhistoryset (function), 1084
positionpresetpositionhistoryget (function), 1084
positionpresetpositionset (subroutine), 1084
positionpresetvelocityset (subroutine), 1084
positionvelocity (function), 1084
positionvelocityisgettable (function), 1085
postprocessor (type), 1234
postprocessors (type), 1234
potential_einasto_scale_free (function), 700
potential_energy_integrand (function), 707
potential_energy_integrand_einasto (function), 700
power_spectra (module), 1261  
power_spectra (program), 601  
Power_Spectra.F90 (file), 600  
power_spectra.tasks.F90 (file), 1158  
power_spectra_nonlinear (module), 1264  
power_spectra_nonlinear_cosmicemu (module), 1263  
power_spectra_nonlinear_linear (module), 1265  
power_spectra_nonlinear_peacockdodds1996 (module), 1265  
power_spectrum (function), 1263  
power_spectrum_close_file (subroutine), 1158  
power_spectrum_compute (subroutine), 1158  
power_spectrum_dimensionless (function), 1263  
power_spectrum_initialize (subroutine), 1266  
power_spectrum_logarithmic_derivative (function), 1263  
power_spectrum_nonlinear (function), 1264  
power_spectrum_nonlinear_cosmicemu (function), 1264  
power_spectrum_nonlinear_cosmicemu_initialize (subroutine), 1264  
power_spectrum_nonlinear_initialize (subroutine), 1264  
power_spectrum_nonlinear_linear (function), 1265  
power_spectrum_nonlinear_linear_initialize (subroutine), 1265  
power_spectrum_nonlinear_peacockdodds1996 (function), 1265  
power_spectrum_nonlinear_peacockdodds1996_initialize (subroutine), 1265  
power_spectrum_open_file (subroutine), 1158  
power_spectrum_output (subroutine), 1158  
power_spectrum_power_law_tabulate (subroutine), 1267  
power_spectrum_tasks (module), 1158  
power_spectrum_window_function (function), 1268  
power_spectrum_window_function_sharp_kspace (function), 1268  
power_spectrum_window_function_th_kss_hybrid (function), 1271  
power_spectrum_window_function_top_hat (function), 1270  
power_spectrum_window_function_wavenumber_maximum (function), 1268  
power_spectrum_window_function_wavenumber_maximum_sharp_kspace (function), 1269  
power_spectrum_window_function_wavenumber_maximum_th_kss_hybrid (function), 1271  
power_spectrum_window_function_wavenumber_maximum_top_hat (function), 1270  
power_spectrum_window_functions (module), 1268  
power_spectrum_window_functions_initialize (subroutine), 1268  
power_spectrum_window_functions_sharp_kspace (module), 1268  
power_spectrum_window_functions_sharp_kspace_initialize (subroutine), 1270  
power_spectrum_window_functions_th_kss_hybrid (module), 1271  
power_spectrum_window_functions_th_kss_hybrid_initialize (subroutine), 1271  
power_spectrum_window_functions_top_hat (module), 1270  
power_spectrum_window_functions_top_hat_initialize (subroutine), 1270  
primordial_power_spectra (module), 1266  
primordial_power_spectra_transferred (module), 1267  
primordial_power_spectrum (function), 1266  
primordial_power_spectrum_logarithmic_derivative (function), 1266  
primordial_power_spectrum_power_law (module), 1266  
primordial_power_spectrum_power_law_initialize (subroutine), 1267  
primordial_power_spectrum_power_law_state_retrieve (subroutine), 1267
CODE INDEX

primordial_power_spectrum_power_law_state_store (subroutine), 1267
primordial_power_spectrum_state_retrieve (subroutine), 1266
primordial_power_spectrum_transferred (function), 1267
progenitor_mass_function (function), 830, 832
pseudo_random (module), 876
pseudo_random_free (subroutine), 877
pseudo_random_get (function), 877
pseudo_random_retrieve (subroutine), 877
pseudo_random_store (subroutine), 877
put (interface), 819
put_ch (subroutine), 819
put_line (interface), 819
put_line_ch (subroutine), 819
put_line_unit_ch (subroutine), 819
put_line_unit_vs (subroutine), 819
put_line_vs (subroutine), 819
put_vs (subroutine), 819
quasi_random (module), 878
quasi_random_free (subroutine), 878
quasi_random_get (interface), 878
quasi_random_get_array (function), 878
quasi_random_get_scalar (function), 878
radiation.cosmic_microwave_background.F90 (file), 1158
radiation.intergalactic_background.F90 (file), 1159
radiation.intergalactic_background.file.F90 (file), 1160
radiation_null.F90 (file), 1160
radiation_cmb (module), 1158
radiation_define (subroutine), 1150
radiation_flux (function), 1151
radiation_flux_cmb (subroutine), 1159
radiation_flux_intergalactic_background (subroutine), 1159
radiation_flux_null (subroutine), 1161
radiation_igb_file (module), 1160
radiation_igb_file_flux (subroutine), 1160
radiation_igb_file_format_version (function), 1160
radiation_igb_file_initialize (subroutine), 1160
radiation_igb_file_set (subroutine), 1160
radiation_initialize_intergalactic_background (subroutine), 1159
radiation_integrate_over_cross_section (function), 1151
radiation_intergalactic_background (module), 1159
radiation_is_defined (function), 1151
radiation_null (module), 1161
radiation_set (subroutine), 1151
radiation_set_cmb (subroutine), 1159
radiation_set_intergalactic_background (subroutine), 1159
radiation_set_null (subroutine), 1161
CODE INDEX

radiation_structure (module), 1149
radiation_temperature (function), 1151
radiation_temperature_cmb (subroutine), 1159
radiation_temperature_intergalactic_background (subroutine), 1160
radiation_temperature_null (subroutine), 1161
radiation_time (function), 1151
radiationdata (type), 1151
radiationstructure (type), 1152
radius_from_specific-angular_momentum_einasto (function), 700
radius_from_specific-angular_momentum_isothermal (function), 711
radius_from_specific-angular_momentum_nfw (function), 707
radius_from_specific-angular_momentum_scale_free (function), 700
radius_from_specific-angular_momentum_table_make (subroutine), 701
radiusperturbation (function), 1272
radiusspecifier (type), 770, 776
ram_pressure_stripping.mass_loss_rate.disks.F90 (file), 1161
ram_pressure_stripping.mass_loss_rate.disks.null.F90 (file), 1161
ram_pressure_stripping.mass_loss_rate.disks.simple.F90 (file), 1162
ram_pressure_stripping.mass_loss_rate_disk (function), 1161
ram_pressure_stripping.mass_loss_rate_disk_null (function), 1162
ram_pressure_stripping.mass_loss_rate_disk_simple (function), 1162
ram_pressure_stripping.mass_loss_rate_disks (module), 1161
ram_pressure_stripping.mass_loss_rate_disks_null (module), 1162
ram_pressure_stripping.mass_loss_rate_disks_null_init (subroutine), 1162
ram_pressure_stripping_mass_loss_rate_disks_simple (module), 1162
ram_pressure_stripping_mass_loss_rate_disks_simple_init (subroutine), 1163
read_node_data (subroutine), 843
read_particle_data (subroutine), 843
recycled_fraction_integrand (function), 1187
redshift_from_expansion_factor (function), 676
remove (interface), 820
remove_ch (function), 820
remove_vs (function), 820
repeat (interface), 820
repeat_ (function), 820
replace (interface), 820
replace_ch (function), 820
replace_ch_ch_auto (function), 820
replace_ch_ch_ch_target (function), 820
replace_ch_ch_fixed (function), 820
replace_ch_ch_vs_target (function), 820
replace_ch_vs (function), 820
replace_ch_vs_ch (function), 820
replace_ch_vs_ch_target (function), 820
replace_ch_vs_vs_target (function), 820
replace_vs (function), 820
replace_vs_ch (function), 820
replace_vs_ch_ch (function), 820
replace_vs_ch_ch_target (function), 820
replace_vs_ch_fixed (function), 820
replace_vs_ch_vs_target (function), 820
replace_vs_vs (function), 820
replace_vs_vs_ch (function), 820
replace_vs_vs_target (function), 820

replace_vs_vs_ch_target (function), 821
replace_vs_vs_fixed (function), 821
replace_vs_vs_vs_target (function), 821
reset_records (subroutine), 851
root_finder (module), 879
root_finder_find (function), 880
root_finder_is_initialized (function), 880
root_finder_range_expand (subroutine), 880
root_finder_root_function (subroutine), 880
root_finder_tolerance (subroutine), 880
root_finder_type (subroutine), 880
root_finder_wrapper_function (function), 880
root_function_1 (function), 1297
root_function_2 (function), 1297
root_function_3 (function), 1297
rootfinder (type), 881
rowstructure (type), 1325

satellite_merger_process (subroutine), 852
satellite_merging_mass_movement (subroutine), 1169
satellite_merging_mass_movement_baugh2005 (subroutine), 1169
satellite_merging_mass_movement_simple (subroutine), 1170
satellite_merging_mass_movement_store (subroutine), 1169
satellite_merging_mass_movement_very_simple (subroutine), 1171
satellite_merging_mass_movements (module), 1169
satellite_merging_mass_movements_baugh2005 (module), 1168
satellite_merging_mass_movements_baugh2005_initialize (subroutine), 1169
satellite_merging_mass_movements_descriptors (module), 1170
satellite_merging_mass_movements_simple (module), 1170
satellite_merging_mass_movements_simple_initialize (subroutine), 1170
satellite_merging_mass_movements_very_simple (module), 1171
satellite_merging_mass_movements_very_simple_initialize (subroutine), 1171
satellite_merging_remnant_progenitor_properties (subroutine), 1175
satellite_merging_remnant_progenitor_properties_cole2000 (subroutine), 1174
satellite_merging_remnant_progenitor_properties_cole2000_init (subroutine), 1174
satellite_merging_remnant_progenitor_properties_standard (subroutine), 1175
satellite_merging_remnant_progenitor_properties_standard_init (subroutine), 1175
satellite_merging_remnant_progenitors_properties_cole2000 (module), 1174
satellite_merging_remnant_progenitors_properties_standard (module), 1175
satellite_merging_remnant_size (subroutine), 1173
satellite_merging_remnant_size_cole2000 (subroutine), 1172
satellite_merging_remnant_size_covington2008 (subroutine), 1172
satellite_merging_remnant_size_null (subroutine), 1173
satellite_merging_remnant_sizes (module), 1173
satellite_merging_remnant_sizes_cole2000 (module), 1172
satellite_merging_remnant_sizes_cole2000_initialize (subroutine), 1172
satellite_merging_remnant_sizes_covington2008 (module), 1172
satellite_merging_remnant_sizes_covington2008_initialize (subroutine), 1173
satellite_merging_remnant_sizes_null (module), 1173
satellite_merging_remnant_sizes_null_initialize (subroutine), 1173
satellite_merging_remnant_sizes_progenitors (module), 1174
satellite_merging_remnant_sizes_properties (module), 1176
satellite_merging_times_preset (module), 1171
satellite_merging_timescales (module), 1176
satellite_merging_timescales_initialize (subroutine), 1176
satellite_move_to_new_host (subroutine), 1181
satellite_orbit_convert_to_current_potential (function), 1180
satellite_orbit_equivalent_circular_orbit_radius (function), 1180
satellite_orbit_pericenter_phase_space_coordinates (subroutine), 1180
satellite_orbits (module), 1179
satellite_promotion (module), 1181
satellite_time_until_merging (function), 1176
satellite_time_until_merging_boylankolchin2008 (function), 1164
satellite_time_until_merging_boylankolchin2008_initialize (subroutine), 1164
satellite_time_until_merging_jiang2008 (function), 1164
satellite_time_until_merging_jiang2008_initialize (subroutine), 1165
satellite_time_until_merging_lacey_cole (function), 1165
satellite_time_until_merging_lacey_cole_initialize (subroutine), 1165
satellite_time_until_merging_lacey_cole_tormen (function), 1167
satellite_time_until_merging_lacey_cole_tormen_initialize (subroutine), 1167
satellite_time_until_merging_lacey_cole_tormen_snapshot (subroutine), 1167
satellite_time_until_merging_lacey_cole_tormen_state_retrieve (subroutine), 1167
satellite_time_until_merging_lacey_cole_tormen_state_store (subroutine), 1167
satellite_time_until_merging_preset (function), 1171
satellite_time_until_merging_preset_initialize (subroutine), 1171
satellite_time_until_merging_wetzel_white (function), 1167
satellite_time_until_merging_wetzel_white_initialize (subroutine), 1168
satelliteboundmass (function), 1085
satelliteboundmasshistory (function), 1085
satelliteboundmasshistoryisgettable (function), 1085
satelliteboundmassisgettable (function), 1085
satellitecountlinked (function), 1085
satellitetrackbyinterrupt (subroutine), 1085
satellitecreateelinked (subroutine), 1085
satelliteedere linked (subroutine), 1085
satelliteget (function), 1086
satellitegettime (function), 1086
satellitegettimeisgettable (function), 1086
satellitenullbindingdouble0inout (function), 1086
satellitenullbindinghistory0inout (function), 1086
satellitenullbindinginteger0in (function), 1086
satellitenullbindingkeplerorbit0inout (function), 1086
satellitenullbindingratedouble0inout (subroutine), 1086
satellitenullbindingratehistory0inout (subroutine), 1086
satellitenullbindingrateinteger0in (subroutine), 1086
satellitenullbindingratekeplerorbit0inout (subroutine), 1087
satellitenullbindingsetdouble0inout (subroutine), 1087
satellitenullbindingsethistory0inout (subroutine), 1087
satellitenullbindingsetinteger0in (subroutine), 1087
satellitenullbindingsetkeplerorbit0inout (subroutine), 1087
satellitepresetboundmasshistoryget (function), 1087
satellitepresetboundmasshistoryset (subroutine), 1087
satellitepresettimeofmergingget (function), 1087
satellitepresettimeofmergingset (subroutine), 1087
satellitepresetvirialorbitget (function), 1088
satellitepresetvirialorbitset (subroutine), 1088
satellites.merging.dynamical_friction.timescale.Lacey-Cole.F90 (file), 1165
satellites.merging.dynamical_friction.timescale.Lacey-Cole_Tormen.F90 (file), 1165
satellites.merging.dynamical_friction.timescale.utilities.F90 (file), 1168
satellites.merging.dynamical_friction.timescale.Wetzel-White.F90 (file), 1167
satellites.merging.mass_movements.descriptors.F90 (file), 1169
satellites.merging.mass_movements.F90 (file), 1169
satellites.merging.mass_movements.simple.F90 (file), 1170
satellites.merging.mass_movements.very_simple.F90 (file), 1170
satellites.merging.preset.F90 (file), 1171
satellites.merging.remnant_sizes.Cole2000.F90 (file), 1171
satellites.merging.remnant_sizes.Covington2008.F90 (file), 1172
satellites.merging.remnant_sizes.F90 (file), 1173
satellites.merging.remnant_sizes.null.F90 (file), 1173
satellites.merging.remnant_sizes.progenitor_properties.Cole2000.F90 (file), 1174
satellites.merging.remnant_sizes.progenitor_properties.F90 (file), 1174
satellites.merging.remnant_sizes.progenitor_properties.standard.F90 (file), 1175
satellites.merging.remnant_sizes.properties.F90 (file), 1175
satellites.merging.timescale.F90 (file), 1176
satellites.merging.virial_orbits.F90 (file), 1177
satellites.merging.virial_orbits.fixed.F90 (file), 1179
satellites.merging.virial_orbits.Wetzel2010.F90 (file), 1178
satellites.orbits.F90 (file), 1179
satellites.promotion.F90 (file), 1180
satelliteteststandardboundmassaccount (function), 1088
satelliteteststandardboundmassget (function), 1088
satelliteteststandardboundmassrate (subroutine), 1088
satelliteteststandardboundmassscale (subroutine), 1088
satelliteteststandardboundmassset (subroutine), 1088
satelliteteststandardmergetimeccount (function), 1088
satelliteteststandardmergetimerate (subroutine), 1088
satelliteteststandardmergetimescale (subroutine), 1089
satelliteteststandardmergetimesset (subroutine), 1089
satelliteteststandardvirialorbitget (function), 1089
satelliteteststandardvirialorbitsetvalue (function), 1089
satelliteteststandardvirialorbitset (subroutine), 1089
satelliteteststandardvirialorbitsetvalue (subroutine), 1089
satellite_timeofmerging (function), 1089
satellite_timeofmergingisgettable (function), 1089
satelliteverysimplemergetimecount (function), 1089
satelliteverysimplemergetimerate (subroutine), 1090
satelliteverysimplemergetimescale (subroutine), 1090
satelliteverysimplemergetimeset (subroutine), 1090
satellitevirialorbit (function), 1090
satellitevirialorbitgetfunction (subroutine), 1090
satellitevirialorbitgetisattached (function), 1090
satellitevirialorbitisgettable (function), 1090
satellitevirialorbitsetfunction (subroutine), 1090
satellitevirialorbitsetisattached (function), 1090
scan (interface), 821
scan_ch_vs (function), 821
scan_for_branch_jumps (subroutine), 843
scan_for_mergers (subroutine), 843
scan_for_subhalo_promotions (subroutine), 844
scan_vs_ch (function), 822
scan_vs_vs (function), 822
search_array (interface), 881
search_array_double (function), 881
search_array_for_closest (function), 881
search_array_integer8 (function), 881
search_array_varstring (function), 882
search_array_double (function), 881
search_array_double_c (subroutine), 883
search_array_integer (subroutine), 883
search_array_integer_c (subroutine), 883
search_index_do (interface), 883
search_index_double (function), 883
serializetoarraycount (function), 1091
serializetoarrayrates (subroutine), 1091
serializetoarrayscales (subroutine), 1091
serializetoarrayvalues (subroutine), 1091
set_integer_scalar_ch (subroutine), 1323
set_integer_scalar_vs (subroutine), 1323
set_memory_prefix (subroutine), 1353
sigma_8 (function), 1263
simple_merger_tree_file_maker (program), 601
Simple_Merger_Tree_File_Maker.F90 (file), 601
sine_integral (function), 825
size_integer_scalar (function), 1323
sneia_cumulative_number (function), 1221
sneia_cumulative_number_nagashima (function), 1222
sneia_cumulative_yield (function), 1221
sneia_cumulative_yield_nagashima (function), 1222
snepii_cumulative_energy (function), 1220
snepii_cumulative_energy_hegerwoosley (function), 1221
solve_for_radius (subroutine), 726, 727, 731, 732
sort (module), 882
sort_do (interface), 882
sort_double (subroutine), 882
sort_double_c (subroutine), 883
sort_integer (subroutine), 883
sort_integer_c (subroutine), 883
sort_index_do (interface), 883
sort_index_double (function), 883
sort_index_double (function), 883
sort_index_do_integer8_c (subroutine), 883
specific-angular-momentum_nfw_scale_free (function), 709
spectratable (type), 1232
spherical_collapse_delta_critical_initialize (subroutine), 1274
spherical_collapse_delta_virial_initialize (subroutine), 1274
spherical_collapse_critical_overdensity (subroutine), 1274
spherical_collapse_dark_energy_critical_overdensity (subroutine), 1272
spherical_collapse_dark_energy_delta_critical_initialize (subroutine), 1272
spherical_collapse_dark_energy_delta_virial_initialize (subroutine), 1272
spherical_collapse_dark_energy_virial_density_contrast (subroutine), 1273
spherical_collapse_matter_dark_energy (module), 1271
spherical_collapse_matter_dark_energy_state_retrieve (subroutine), 1273
spherical_collapse_matter_dark_energy_state_store (subroutine), 1273
spherical_collapse_matter_lambda (module), 1273
spherical_collapse_matter_lambda_state_retrieve (subroutine), 1274
spherical_collapse_matter_lambda_state_store (subroutine), 1274
spherical_collapse_virial_density_contrast (subroutine), 1274
spherical_shell_solid_angle_in_cylcinder (function), 776
spheroidabundancesgas (function), 1091
spheroidabundancesgasisgettable (function), 1091
spheroidabundancesgasrate (subroutine), 1091
spheroidabundancesstellar (function), 1091
spheroidabundancesstellarisgettable (function), 1092
spheroidabundancesstellarrate (subroutine), 1092
spheroidangularmomentum (function), 1092
spheroidangularmomentumisgettable (function), 1092
spheroidangularmomentumrate (subroutine), 1092
spheroidcountlinked (function), 1092
spheroidcreatebyinterrupt (subroutine), 1092
spheroidcreatelinked (subroutine), 1092
spheroiddestroylinked (subroutine), 1093
spheroidenergygasinput (function), 1093
spheroidenergygasinputisgettable (function), 1093
spheroidenergygasinputratefunction (subroutine), 1093
spheroidenergygasinputrateisattached (function), 1093
spheroidget (function), 1093
spheroidhalfmassradius (function), 1093
spheroidhalfmassradiusisgettable (function), 1093
spheroidinitialized (function), 1093
spheroidinitializedisgettable (function), 1094
spheroidluminositiesstellar (function), 1094
spheroidluminositiesstellarisgettable (function), 1094
spheroidluminositiesstellarrate (subroutine), 1094
spheroidmassgas (function), 1094
spheroidmassgasisgettable (function), 1094
spheroidmassgasrate (subroutine), 1094
spheroidmassgassink (function), 1094
spheroidmassgassinkisgettable (function), 1095
spheroidmassgassinkratefunction (subroutine), 1095
spheroidmassgassinkrateisattached (function), 1095
spheroidmassstellar (function), 1095
spheroidmassstellarisgettable (function), 1095
spheroidmassstellarrate (subroutine), 1095
spheroidnullbindingabundances0inout (function), 1095
spheroidnullbindingdouble0inout (function), 1095
spheroidnullbindingdouble1inout (function), 1095
spheroidnullbindinghistory0inout (function), 1096
spheroidnullbindinginteger0in (function), 1096
spheroidnullbindinglogical0inout (function), 1096
spheroidnullbindingrateabundances0inout (subroutine), 1096
spheroidnullbindingratedouble0inout (subroutine), 1096
spheroidnullbindingratedouble1inout (subroutine), 1096
spheroidnullbindingratehistory0inout (subroutine), 1096
spheroidnullbindingrateinteger0in (subroutine), 1096
spheroidnullbindingratelogical0inout (subroutine), 1096
spheroidnullbindingsetabundances0inout (subroutine), 1096
spheroidnullbindingsetdouble0inout (subroutine), 1097
spheroidnullbindingsetdouble1inout (subroutine), 1097
spheroidnullbindingsethistory0inout (subroutine), 1097
spheroidnullbindingsetinteger0in (subroutine), 1097
spheroidnullbindingsetlogical0inout (subroutine), 1097
spheroidradius (function), 1097
spheroidradiusisgettable (function), 1097
spheroidstandardabundancesgascount (function), 1097
spheroidstandardabundancesgasget (function), 1097
spheroidstandardabundancesgasrate (subroutine), 1097
spheroidstandardabundancesgasrategeneric (subroutine), 1098
spheroidstandardabundancesgasscale (subroutine), 1098
spheroidstandardabundancesgasset (subroutine), 1098
spheroidstandardabundancesstellarcount (function), 1098
spheroidstandardabundancesstellarset (subroutine), 1098
spheroidstandardabundancesstellarrate (subroutine), 1098
spheroidstandardabundancesstellarrategeneric (subroutine), 1098
spheroidstandardabundancesstellarscale (subroutine), 1098
spheroidstandardabundancesstellarset (subroutine), 1099
spheroidstandardangularmomentumcount (function), 1099
spheroidstandardangularmomentumget (function), 1099
spheroidstandardangularmomentumrate (subroutine), 1099
spheroidstandardangularmomentumrategeneric (subroutine), 1099
spheroidstandardangularmomentumscale (subroutine), 1099
spheroidstandardangularmomentumset (subroutine), 1099
spheroidstandardcreatetimefunctionset (subroutine), 1099
spheroidstandardenergygasinputrate (function), 1100
spheroidstandardisinitializedget (function), 1100
spheroidstandardisinitializedset (subroutine), 1100
spheroidstandardluminositiesstellarcount (function), 1100
spheroidstandardluminositiesstellarset (function), 1100
spheroidstandardluminositiesstellarrate (subroutine), 1100
spheroidstandardluminositiesstellarrategeneric (subroutine), 1100
spheroidstandardluminositiesstellarscale (subroutine), 1100
spheroidstandardluminositiesstellarset (subroutine), 1101
spheroidstandardmassgascount (function), 1101
spheroidstandardmassgasget (function), 1101
spheroidstandardmassgasrate (subroutine), 1101
spheroidstandardmassgasrategeneric (subroutine), 1101
spheroidstandardmassgasscale (subroutine), 1101
spheroidstandardmassgasset (subroutine), 1101
spheroidstandardmassgassinkrate (subroutine), 1101
spheroidstandardmassstellarcnt (function), 1101
spheroidstandardmassstellarget (function), 1102
spheroidstandardmassstellarrate (subroutine), 1102
spheroidstandardmassstellarrategeneric (subroutine), 1102
spheroidstandardmassstellarscale (subroutine), 1102
spheroidstandardmassstellarset (subroutine), 1102
spheroidstandardradiusget (function), 1102
spheroidstandardradiusset (subroutine), 1102
spheroidstandardstarformationhistorycount (function), 1102
spheroidstandardstarformationhistoryget (function), 1103
spheroidstandardstarformationhistoryrate (subroutine), 1103
spheroidstandardstarformationhistoryrategeneric (subroutine), 1103
spheroidstandardstarformationhistoryscale (subroutine), 1103
spheroidstandardstarformationhistoryset (subroutine), 1103
spheroidstandardstarformationrateget (function), 1103
spheroidstandardstellarpropertieshistorycount (function), 1103
spheroidstandardstellarpropertieshistoryget (function), 1103
spheroidstandardstellarpropertieshistoryrate (subroutine), 1104
spheroidstandardstellarpropertieshistoryrategeneric (subroutine), 1104
spheroidstandardstellarpropertieshistoryscale (subroutine), 1104
spheroidstandardstellarpropertieshistoryset (subroutine), 1104
spheroidstandardvelocityget (function), 1104
spheroidstandardvelocityset (subroutine), 1104
spheroidstarformationhistory (function), 1104
spheroidstarformationhistoryisgettable (function), 1104
spheroidstarformationhistoryrate (subroutine), 1105
spheroidstarformationrate (function), 1105
spheroidstarformationrategetfunction (subroutine), 1105
spheroidstarformationrateisgettable (function), 1105
spheroidstellarpropertieshistory (function), 1105
spheroidstellarpropertieshistoryisgettable (function), 1105
spheroidstellarpropertieshistoryrate (subroutine), 1105
spheroidvelocity (function), 1106
spheroidvelocityisgettable (function), 1106
spincountlinked (function), 1106
spincreatebyinterrupt (subroutine), 1106
spincreatelinked (subroutine), 1106
spindestroylinked (subroutine), 1106
spinget (function), 1106
spinnullbindingdouble0inout (function), 1106
spinnullbindinginteger0in (function), 1106
spinnullbindingratedouble0inout (subroutine), 1107
spinnullbindingrateinteger0in (subroutine), 1107
spinnullbindingsetdouble0inout (subroutine), 1107
spinnullbindingsetinteger0in (subroutine), 1107
spinpresetspincount (function), 1107
spinpresetspinget (function), 1107
spinpresetspingleadrateget (function), 1107
spinpresetspingleadrateset (subroutine), 1107
spinpresetspinscale (subroutine), 1108
spinpresetspinset (subroutine), 1108
spinrandomspincount (function), 1108
spinrandomspinget (function), 1108
spinrandomspingleadrate (subroutine), 1108
spinrandomspingescale (subroutine), 1108
spinrandomspinset (subroutine), 1108
spinspin (function), 1108
spinspingrowthrate (function), 1108
spinspingrowthratesetgettable (function), 1109
spinspinisgettable (function), 1109
split (interface), 822
split_ch (subroutine), 822
split_vs (subroutine), 822
spmpar (function), 646
star_ejected_mass (function), 1217
star_ejected_mass_file (function), 1219
star_formation.feedback.disks.Creasey2012.F90 (file), 1197
star_formation.feedback.disks.F90 (file), 1198
star_formation.feedback.disks.fixed.F90 (file), 1199
star_formation.feedback.disks.halo_scaling.F90 (file), 1199
star_formation.feedback.disks.power_law.F90 (file), 1200
star_formation.feedback.spheroids.F90 (file), 1201
star_formation.feedback.spheroids.power_law.F90 (file), 1201
star_formation.feedback.expulsion.disks.F90 (file), 1202
star_formation.feedback.expulsion.disks.null.F90 (file), 1203
star_formation.feedback.expulsion.disks.superwind.F90 (file), 1203
star_formation.feedback.expulsion.spheroids.F90 (file), 1205
star_formation.feedback.expulsion.spheroids.null.F90 (file), 1205
star_formation.feedback.expulsion.spheroids.superwind.F90 (file), 1206
star_formation.IMF.Baugh2005TopHeavy.F90 (file), 1181
star_formation.IMF.Chabrier.F90 (file), 1182
star_formation.IMF.F90 (file), 1183
star_formation.IMF.Kennicutt.F90 (file), 1187
star_formation.IMF.Kroupa.F90 (file), 1189
star_formation.IMF.Miller-Scalo.F90 (file), 1190
star_formation.IMF.piecewise_power_law.F90 (file), 1194
star_formation.IMF.Salpeter.F90 (file), 1191
star_formation.IMF.Scalo.F90 (file), 1192
star_formation.IMF.select.diskSpheroid.F90 (file), 1195
star_formation.IMF.select.fixed.F90 (file), 1195
star_formation.IMF.utilities.F90 (file), 1196
star_formation.IMF.utilities.piecewise_power_laws.F90 (file), 1196
star_formation.rate_surface_density.disks.Blitz-Rosolowsky.F90 (file), 1206
star_formation.rate_surface_density.disks.extended_Schmidt.F90 (file), 1210
star_formation.rate_surface_density.disks.F90 (file), 1207
star_formation.rate_surface_density.disks.Kennicutt-Schmidt.F90 (file), 1209
star_formation.rate_surface_density.disks.KMT09.F90 (file), 1208
star_formation.timescales.disks.Baugh2005.cpp (file), 1210
star_formation.timescales.disks.dynamical_time.F90 (file), 1211
star_formation.timescales.disks.F90 (file), 1211
star_formation.timescales.disks.fixed.F90 (file), 1213
star_formation.timescales.disks.halo_scaling.F90 (file), 1213
star_formation.timescales.disks.integrated_surface_density.F90 (file), 1214
star_formation.timescales.spheroids.dynamical_time.F90 (file), 1215
star_formation.timescales.spheroids.F90 (file), 1214
star_formation_expulsive_feedback_disk_outflow_rate (function), 1202
star_formation_expulsive_feedback_disk_outflow_rate_null (function), 1203
star_formation_expulsive_feedback_disk_outflow_rate_sw (function), 1203
star_formation_expulsive_feedback_disks_initialize (subroutine), 1202
star_formation_expulsive_feedback_disks_null (module), 1203
star_formation_expulsive_feedback_disks_null_initialize (subroutine), 1203
star_formation_expulsive_feedback_disks_superwind (module), 1203
star_formation_expulsive_feedback_disks_sw_initialize (subroutine), 1204
star_formation_expulsive_feedback_spheroid_outflow_rate (function), 1205
star_formation_expulsive_feedback_spheroid_outflow_rate_null (function), 1205
star_formation_expulsive_feedback_spheroid_outflow_rate_sw (function), 1206
star_formation_expulsive_feedback_spheroids_initialize (subroutine), 1205
star_formation_expulsive_feedback_spheroids_null (module), 1205
star_formation_expulsive_feedback_spheroids_null_initialize (subroutine), 1206
star_formation_expulsive_feedback_spheroids_superwind (module), 1206
star_formation_expulsive_feedback_spheroids_sw_initialize (subroutine), 1206
star_formation_feedback_disk_outflow_rate (function), 1198
star_formation_feedback_disk_outflow_rate_creesey2012 (function), 1197
star_formation_feedback_disk_outflow_rate_creesey2012_integrand (function), 1198
star_formation_feedback_disk_outflow_rate_fixed (function), 1199
star_formation_feedback_disk_outflow_rate_halo_scaling (function), 1200
star_formation_feedback_disk_outflow_rate_power_law (function), 1200
star_formation_feedback_disk_outflow_rate_template (function), 1199
star_formation_feedback_disk_outflow_rate_template (interface), 1198
star_formation_feedback_disks (module), 1198
star_formation_feedback_disks_creesey2012 (module), 1197
star_formation_feedback_disks_creesey2012_initialize (subroutine), 1198
star_formation_feedback_disks_fixed (module), 1199
star_formation_feedback_disks_fixed_initialize (subroutine), 1199
star_formation_feedback_disks_halo_scaling (module), 1199
star_formation_feedback_disks_halo_scaling_initialize (subroutine), 1200
star_formation_feedback_disks_initialize (subroutine), 1199
star_formation_feedback_disks_power_law (module), 1200
star_formation_feedback_disks_power_law_initialize (subroutine), 1200
star_formation_feedback_expulsion_disks (module), 1202
star_formation_feedback_expulsion_spheroids (module), 1205
star_formation_feedback_spheroid_outflow_rate (function), 1201
star_formation_feedback_spheroid_outflow_rate_power_law (function), 1202
star_formation_feedback_spheroid_outflow_rate_template (function), 1201
star_formation_feedback_spheroid_outflow_rate_template (interface), 1201
star_formation_feedback_spheroids (module), 1201
star_formation_feedback_spheroids_initialize (subroutine), 1201
star_formation_feedback_spheroids_power_law (module), 1201
star_formation_histories_metallicity_split (module), 772
star_formation_histories_metallicity_split_initialize (subroutine), 772
star_formation_histories_null (module), 773
star_formation_histories_null_initialize (subroutine), 773
star_formation_history_create (subroutine), 771
star_formation_history_create_metallicity_split (subroutine), 772
star_formation_history_create_null (subroutine), 773
star_formation_history_metallicity_split_make_history (subroutine), 772
star_formation_history_output (subroutine), 771
star_formation_history_output_metallicity_split (subroutine), 772
star_formation_history_output_null (subroutine), 773
star_formation_history_record (subroutine), 771
star_formation_history_record_metallicity_split (subroutine), 773
star_formation_history_record_null (subroutine), 774
star_formation_history_scales (subroutine), 772
star_formation_history_scales_metallicity_split (subroutine), 773
star_formation_history_scales_null (subroutine), 774
star_formation_imf (module), 1183
star_formation_imf_baugh2005topheavy (module), 1181
star_formation_imf_chabrier (module), 1182
star_formation_imf_initialize (subroutine), 1187
star_formation_imf_initialize_baugh2005topheavy (subroutine), 1181
star_formation_imf_initialize_chabrier (subroutine), 1182
star_formation_imf_initialize_kennicutt (subroutine), 1188
star_formation_imf_initialize_kroupa (subroutine), 1189
star_formation_imf_initialize_millerscalo (subroutine), 1190
star_formation_imf_initialize_piecewisepowerlaw (subroutine), 1194
star_formation_imf_initialize_salpeter (subroutine), 1191
star_formation_imf_initialize_scalo (subroutine), 1193
star_formation_imf_kennicutt (module), 1188
star_formation_imf_kroupa (module), 1189
star_formation_imf_label (function), 1337
star_formation_imf_maximum_mass_baugh2005topheavy (subroutine), 1181
star_formation_imf_maximum_mass_chabrier (subroutine), 1183
star_formation_imf_maximum_mass_kennicutt (subroutine), 1188
CODE INDEX

star_formation_imf_maximum_mass_kroupa (subroutine), 1189
star_formation_imf_maximum_mass_millerscalo (subroutine), 1190
star_formation_imf_maximum_mass_piecewisepowerlaw (subroutine), 1194
star_formation_imf_maximum_mass_salpeter (subroutine), 1191
star_formation_imf_maximum_mass_scalo (subroutine), 1193
star_formation_imf_millerscalo (module), 1190
star_formation_imf_minimum_mass_baugh2005topheavy (subroutine), 1181
star_formation_imf_minimum_mass_chabrier (subroutine), 1183
star_formation_imf_minimum_mass_kennicutt (subroutine), 1188
star_formation_imf_minimum_mass_kroupa (subroutine), 1189
star_formation_imf_minimum_mass_piecewisepowerlaw (subroutine), 1194
star_formation_imf_minimum_mass_salpeter (subroutine), 1192
star_formation_imf_minimum_mass_scalo (subroutine), 1193
star_formation_imf_phi_baugh2005topheavy (subroutine), 1181
star_formation_imf_phi_chabrier (subroutine), 1183
star_formation_imf_phi_kennicutt (subroutine), 1188
star_formation_imf_phi_kroupa (subroutine), 1189
star_formation_imf_phi_millerscalo (subroutine), 1190
star_formation_imf_phi_piecewisepowerlaw (subroutine), 1194
star_formation_imf_phi_salpeter (subroutine), 1192
star_formation_imf_phi_scalo (subroutine), 1193
star_formation_imf_piecewisepowerlaw (module), 1194
star_formation_imf_ppl (module), 1196
star_formation_imf_recycled_instantaneous_baugh2005topheavy (subroutine), 1182
star_formation_imf_recycled_instantaneous_chabrier (subroutine), 1183
star_formation_imf_recycled_instantaneous_kennicutt (subroutine), 1188
star_formation_imf_recycled_instantaneous_kroupa (subroutine), 1189
star_formation_imf_recycled_instantaneous_millerscalo (subroutine), 1191
star_formation_imf_recycled_instantaneous_piecewisepowerlaw (subroutine), 1194
star_formation_imf_recycled_instantaneous_salpeter (subroutine), 1192
star_formation_imf_recycled_instantaneous_scalo (subroutine), 1193
star_formation_imf_register_baugh2005topheavy (subroutine), 1182
star_formation_imf_register_chabrier (subroutine), 1183
star_formation_imf_register_kennicutt (subroutine), 1188
star_formation_imf_register_kroupa (subroutine), 1189
star_formation_imf_register_millerscalo (subroutine), 1191
star_formation_imf_register_name_baugh2005topheavy (subroutine), 1182
star_formation_imf_register_name_chabrier (subroutine), 1183
star_formation_imf_register_name_kennicutt (subroutine), 1188
star_formation_imf_register_name_kroupa (subroutine), 1190
star_formation_imf_register_name_millerscalo (subroutine), 1191
star_formation_imf_register_name_piecewisepowerlaw (subroutine), 1194
star_formation_imf_register_name_salpeter (subroutine), 1192
star_formation_imf_register_name_scalo (subroutine), 1193
star_formation_imf_register_piecewisepowerlaw (subroutine), 1194
star_formation_imf_register_salpeter (subroutine), 1192
star_formation_imf_register_scalo (subroutine), 1193
star_formation_imf_salpeter (module), 1192

1512
<table>
<thead>
<tr>
<th>Code Index</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>star_formation_imf_scalo (module), 1192</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_select_disk_spheroid (module), 1195</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_select_fixed (module), 1195</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_tabulate_baugh2005topheavy (subroutine), 1182</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_tabulate_chabrier (subroutine), 1183</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_tabulate_kennicutt (subroutine), 1188</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_tabulate_kroupa (subroutine), 1190</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_tabulate_millerscalo (subroutine), 1191</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_tabulate_piecewisepowerlaw (subroutine), 1195</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_tabulate_salpeter (subroutine), 1192</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_tabulate_scalo (subroutine), 1193</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_utilities (module), 1196</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_yield_instantaneous_baugh2005topheavy (subroutine), 1182</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_yield_instantaneous_chabrier (subroutine), 1183</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_yield_instantaneous_kennicutt (subroutine), 1189</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_yield_instantaneous_kroupa (subroutine), 1190</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_yield_instantaneous_millerscalo (subroutine), 1191</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_yield_instantaneous_piecewisepowerlaw (subroutine), 1195</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_yield_instantaneous_salpeter (subroutine), 1192</td>
<td></td>
</tr>
<tr>
<td>star_formation_imf_yield_instantaneous_scalo (subroutine), 1193</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_integrand_surface_density (function), 1214</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disk (function), 1207</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disk_br (function), 1207</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disk_exschmidt (function), 1210</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disk_kmt09 (function), 1208</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disk_ks (function), 1209</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disks (module), 1207</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disks_br (module), 1206</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disks_br_initialize (subroutine), 1207</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disks_br_reset (subroutine), 1207</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disks_exschmidt (module), 1210</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disks_exschmidt_initialize (subroutine), 1210</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disks_exschmidt_reset (subroutine), 1210</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disks_initialize (subroutine), 1207</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disks_kmt09 (module), 1208</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disks_kmt09_initialize (subroutine), 1208</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disks_kmt09_reset (subroutine), 1209</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disks_ks (module), 1209</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disks_ks_initialize (subroutine), 1209</td>
<td></td>
</tr>
<tr>
<td>star_formation_rate_surface_density_disks_ks_reset (subroutine), 1210</td>
<td></td>
</tr>
<tr>
<td>star_formation_timescale_disk (function), 1211</td>
<td></td>
</tr>
<tr>
<td>star_formation_timescale_disk_dynamical_time (function), 1211</td>
<td></td>
</tr>
<tr>
<td>star_formation_timescale_disk_fixed (function), 1213</td>
<td></td>
</tr>
<tr>
<td>star_formation_timescale_disk_halo_scaling (function), 1213</td>
<td></td>
</tr>
<tr>
<td>star_formation_timescale_disk_integrated_sd (function), 1214</td>
<td></td>
</tr>
<tr>
<td>star_formation_timescale_disks_dynamical_time (module), 1211</td>
<td></td>
</tr>
<tr>
<td>star_formation_timescale_disks_dynamical_time_initialize (subroutine), 1212</td>
<td></td>
</tr>
<tr>
<td>star_formation_timescale_disks_fixed (module), 1213</td>
<td></td>
</tr>
<tr>
<td>star_formation_timescale_disks_fixed_initialize (subroutine), 1213</td>
<td></td>
</tr>
</tbody>
</table>
star_formation_timescale_disks_halo_scaling (module), 1213
star_formation_timescale_disks_halo_scaling_initialize (subroutine), 1214
star_formation_timescale_disks_halo_scaling_reset (subroutine), 1214
star_formation_timescale_disks_initialize (subroutine), 1211
star_formation_timescale_disks_integrated_sd (module), 1214
star_formation_timescale_disks_integrated_sd_initialize (subroutine), 1214
star_formation_timescale_spheroid (function), 1215
star_formation_timescale_spheroid_dynamical_time (function), 1215
star_formation_timescale_spheroids_dynamical_time (module), 1215
star_formation_timescale_spheroids_dynamical_time_initialize (subroutine), 1216
star_formation_timescale_spheroids_initialize (subroutine), 1215
star_formation_timescales_disks (module), 1211
star_formation_timescales_spheroids (module), 1215
star_initial_mass (function), 1217
star_initial_mass_file (function), 1219
star_is_evolved (function), 1187
star_lifetime (function), 1217
star_lifetime_file (function), 1219
star_metal_yield_mass (function), 1217
star_metal_yield_mass_file (function), 1219
state_initialize (subroutine), 780
stellar_astrophysics (module), 1217
stellar_astrophysics.F90 (file), 1217
stellar_astrophysics.feedback.F90 (file), 1217
stellar_astrophysics.feedback.standard.F90 (file), 1218
stellar_astrophysics.file.F90 (file), 1219
stellar_astrophysics.supernovae.PopulationIII.F90 (file), 1220
stellar_astrophysics.supernovae.PopulationIII.HegerWoosley.F90 (file), 1220
stellar_astrophysics.supernovae_type_Ia.F90 (file), 1221
stellar_astrophysics.supernovae_type_Ia.Nagashima.F90 (file), 1222
stellar_astrophysics.tracks.F90 (file), 1222
stellar_astrophysics.tracks.file.F90 (file), 1223
stellar_astrophysics.winds.F90 (file), 1224
stellar_astrophysics.winds.Leitherer1992.F90 (file), 1224
stellar_astrophysics_file (module), 1219
stellar_astrophysics_file_format_version (function), 1219
stellar_astrophysics_file_initialize (subroutine), 1220
stellar_astrophysics_initialize (subroutine), 1217
stellar_astrophysics_tracks (module), 1222
stellar_astrophysics_tracks_file (module), 1223
stellar_astrophysics_winds (module), 1224
stellar_astrophysics_winds_leitherer1992 (module), 1225
stellar_effective_temperature (function), 1223
stellar_effective_temperature_file (function), 1223
stellar_feedback (module), 1218
stellar_feedback_cumulative_energy_input (function), 1218
stellar_feedback_cumulative_energy_input_standard (function), 1218
stellar_feedback_initialize (subroutine), 1218
stellar_feedback_standard (module), 1218
stellar_feedback_standard_initialize (subroutine), 1219
stellar_luminosity (function), 1223
stellar_luminosity_file (function), 1223
stellar_mass_function_integrand (function), 600
stellar_population_luminosities (module), 1225
stellar_population_luminosities_count (function), 1229
stellar_population_luminosities_get (function), 1229
stellar_population_luminosities_index (function), 1229
stellar_population_luminosities_name (function), 1229
stellar_population_luminosities_output (function), 1229
stellar_population_luminosities_output_count (function), 1229
stellar_population_luminosity (function), 1226
stellar_population_properties (module), 1226
stellar_population_properties_history_count (function), 1226
stellar_population_properties_history_count_instantaneous (function), 1227
stellar_population_properties_history_count_noninstantaneous (function), 1230
stellar_population_properties_history_create (subroutine), 1226
stellar_population_properties_history_create_instantaneous (subroutine), 1227
stellar_population_properties_history_create_noninstantaneous (subroutine), 1230
stellar_population_properties_instantaneous (module), 1227
stellar_population_properties_instantaneous_initialize (subroutine), 1227
stellar_population_properties_luminosities (module), 1228
stellar_population_properties_luminosities_initialize (subroutine), 1229
stellar_population_properties_noninstantaneous (module), 1230
stellar_population_properties_noninstantaneous_initialize (subroutine), 1230
stellar_population_properties_rates (subroutine), 1227
stellar_population_properties_rates_initialize (subroutine), 1227
stellar_population_properties_rates_instantaneous (subroutine), 1228
stellar_population_properties_rates_noninstantaneous (subroutine), 1230
stellar_population_properties_scales (subroutine), 1227
stellar_population_properties_scales_instantaneous (subroutine), 1228
stellar_population_properties_scales_noninstantaneous (subroutine), 1230
stellar_population_spectra (module), 1231
stellar_population_spectra_conroy (module), 1231
stellar_population_spectra_conroy_get (function), 1231
stellar_population_spectra_conroy_initialize (subroutine), 1231
stellar_population_spectra_conroy_initialize_imf (subroutine), 1231
stellar_population_spectra_file (module), 1232
stellar_population_spectra_file_format_current (function), 1232
stellar_population_spectra_file_get (function), 1232
stellar_population_spectra_file_initialize (subroutine), 1233
stellar_population_spectra_file_initialize_imf (subroutine), 1233
stellar_population_spectra_file_interpolate (function), 1233
stellar_population_spectra_file_read (subroutine), 1233
stellar_population_spectra_file_tabulation (subroutine), 1233
stellar_population_spectra_postprocess (module), 1233
stellar_population_spectra_postprocess_identity (subroutine), 1236
stellar_population_spectra_postprocess_identity_init (subroutine), 1236
stellar_population_spectra_postprocess_lyc_suppress (subroutine), 1234
string_c_to_fortran (function), 1357
string_count_words (function), 1357
string_handling (module), 1353
string_join (function), 1357
string_levenshtein_distance (function), 1357
string_lower_case (function), 1357
string_split_words (interface), 1358
string_split_words_char (subroutine), 1358
string_split_words_varstring (subroutine), 1358
string_subscript (function), 1358
string_superscript (function), 1358
string_upper_case (function), 1358
string_upper_case_first (function), 1358
structure_formation.cosmological_mass_variance.filtered_power_spectrum.F90 (file), 1237
structure_formation.critical_overdensity.F90 (file), 1238
structure_formation.critical_overdensity.mass_scaling.null.F90 (file), 1241
structure_formation.critical_overdensity.mass_scaling.warm_dark_matter.F90 (file), 1241
structure_formation.excursion_sets.barrier.critical_overdensity.F90 (file), 1243
structure_formation.excursion_sets.barrier.F90 (file), 1242
structure_formation.excursion_sets.barrier.linear.F90 (file), 1244
structure_formation.excursion_sets.barrier.quadratic.F90 (file), 1244
structure_formation.excursion_sets.barrier.remap.null.F90 (file), 1246
structure_formation.excursion_sets.barrier.remap.scale.F90 (file), 1246
structure_formation.excursion_sets.barrier.remap.Sheth-Mo-Tormen.F90 (file), 1245
structure_formation.excursion_sets.first_crossing_distribution.F90 (file), 1247
structure_formation.excursion_sets.first_crossing_distribution.Farahi.F90 (file), 1248
structure_formation.excursion_sets.first_crossing_distribution.linear_barrier.F90 (file), 1252
structure_formation.excursion_sets.first_crossing_distribution.Zhang_Hui.F90 (file), 1249
structure_formation.excursion_sets.first_crossing_distribution.Zhang_Hui.utilities.F90 (file),
structure_formation.excursion_sets.first_crossing_distribution.Zhang_Hui_high_order.F90 (file), 1251
structure_formation.halo_bias.F90 (file), 1253
structure_formation.halo_bias.Press-Schechter.F90 (file), 1254
structure_formation.halo_bias.SMT.F90 (file), 1254
structure_formation.halo_bias.Tinker2010.F90 (file), 1255
structure_formation.halo_mass_function.F90 (file), 1256
structure_formation.halo_mass_function.Press-Schechter.F90 (file), 1257
structure_formation.halo_mass_function.Sheth-Tormen.F90 (file), 1257
structure_formation.halo_mass_function.Tinker2008.F90 (file), 1258
structure_formation.linear_growth.F90 (file), 1258
structure_formation.linear_growth.simple.F90 (file), 1259
structure_formation.power_spectrum.F90 (file), 1261
structure_formation.power_spectrum.nonlinear.CosmicEmu.F90 (file), 1263
structure_formation.power_spectrum.nonlinear.F90 (file), 1264
structure_formation.power_spectrum.nonlinear.linear.F90 (file), 1265
structure_formation.power_spectrum.nonlinear.PeacockDodds1996.F90 (file), 1264
structure_formation.power_spectrum.primordial.F90 (file), 1266
structure_formation.power_spectrum.primordial.power_law.F90 (file), 1266
structure_formation.power_spectrum.primordial.transferred.F90 (file), 1267
structure_formation.power_spectrum.variance.window_function.F90 (file), 1267
structure_formation.power_spectrum.variance.window_function.sharp_kSpace.F90 (file), 1268
structure_formation.power_spectrum.variance.window_function.top_hat.F90 (file), 1270
structure_formation.power_spectrum.variance.window_function.top_hat_kspace_sharp_hybrid.F90 (file), 1270
structure_formation.sphericalCollapse.matter_dark_energy.F90 (file), 1271
structure_formation.sphericalCollapse.matter_lambda.F90 (file), 1273
structure_formation.transfer_function.BBKS.F90 (file), 1275
structure_formation.transfer_function.CMBFast.F90 (file), 1275
structure_formation.transfer_function.Eisenstein_Hu.F90 (file), 1276
structure_formation.transfer_function.F90 (file), 1277
structure_formation.transfer_function.file.F90 (file), 1278
structure_formation.transfer_function.null.F90 (file), 1278
structure_formation.virial_density_contrast.Bryan_Norman.F90 (file), 1279
structure_formation.virial_density_contrast.F90 (file), 1279
structure_formation.virial_density_contrast.fixed.F90 (file), 1281
structure_formation.virial_density_contrast.Kitayama_Suto1996.F90 (file), 1281
subresoultion_fraction_integrand_generalized (function), 831
supernovae_population_iii (module), 1220
supernovae_population_iii_hegerwoosley (module), 1220
supernovae_population_iii_hegerwoosley_initialize (subroutine), 1221
supernovae_population_iii_initialize (subroutine), 1220
supernovae_type_ia (module), 1221
supernovae_type_ia_initialize (subroutine), 1221
supernovae_type_ia_nagashima (module), 1222
supernovae_type_ia_nagashima_initialize (subroutine), 1222
system.command.F90 (file), 1282
system.load.F90 (file), 1282
system_command (module), 1282
system_command_do (subroutine), 1282
system_load (module), 1282
system_load_get (subroutine), 1282
system_processor_count (function), 1282
t_cmb (function), 686
table (type), 1153
tableid (type), 1153
tableid_find_effective_x (function), 1153
tableid_is_monotonic (function), 1153
tableid_size (function), 1153
tableid_x (function), 1153
tableid_xs (function), 1153
tableid_y (function), 1153
tableid_ys (function), 1153
tableidgeneric (type), 1154
tableidlinearcspline (type), 1154
tableidlinearlinear (type), 1154
tableidlogarithmiccspline (type), 1154
table1dlogarithmiclinear (type), 1154
table_1d_destroy (subroutine), 1154
table_1d_reverse (subroutine), 1154
table_generic_1d_create (subroutine), 1154
table_generic_1d_destroy (subroutine), 1154
table_generic_1d_interpolate (function), 1155
table_generic_1d_interpolate_gradient (function), 1155
table_generic_1d_populate (subroutine), 1155
table_generic_1d_populate_single (subroutine), 1155
table_linear_1d_create (subroutine), 1155
table_linear_1d_interpolate (function), 1155
table_linear_1d_interpolate_gradient (function), 1155
table_linear_1d_populate (subroutine), 1155
table_linear_1d_populate_single (subroutine), 1156
table_linear_cspline_1d_coefficients (subroutine), 1156
table_linear_cspline_1d_compute_spline (subroutine), 1156
table_linear_cspline_1d_create (subroutine), 1156
table_linear_cspline_1d_destroy (subroutine), 1156
table_linear_cspline_1d_interpolate (function), 1156
table_linear_cspline_1d_interpolate_gradient (function), 1156
table_linear_cspline_1d_populate (subroutine), 1156
table_linear_cspline_1d_populate_single (subroutine), 1156
table_logarithmic_1d_create (subroutine), 1157
table_logarithmic_1d_interpolate (function), 1157
table_logarithmic_1d_interpolate_gradient (function), 1157
table_logarithmic_1d_x (function), 1157
table_logarithmic_1d_xs (function), 1157
table_logarithmic_cspline_1d_create (subroutine), 1157
table_logarithmic_cspline_1d_interpolate (function), 1157
table_logarithmic_cspline_1d_interpolate_gradient (function), 1157
table_logarithmic_cspline_1d_x (function), 1157
table_logarithmic_cspline_1d_xs (function), 1157
tables (module), 1152
tcollapse (function), 1274
test_abundances (program), 1286
test_array_monotonicity (program), 1287
test_black_hole_fundamentals (program), 1287

test_comparison (program), 1288

test_coordinate_systems (program), 1291

test_hashes (program), 1291

test_hashes_cryptographic (program), 1292

test_integration (program), 1292

test_integration_functions (module), 1292

test_interpolation (program), 1293

test_interpolation_2d (program), 1293

test_make_ranges (program), 1295

test_mass_distributions (program), 1295

test_math_special_functions (program), 1295

test_meshes (program), 1295
test_nfw96_concentration_dark_energy (program), 1283

test_node_task (subroutine), 1296

test_nodes (program), 1296

test_nodes_tasks (module), 1296

test_ode_solver (program), 1283

test_ode_solver_functions (module), 1284

test_perfect_hashes (program), 1292

test_prada2011_concentration (program), 1284

test_root_finding (program), 1297

test_root_finding_functions (module), 1297

test_search (program), 1297

test_sort (program), 1298

test_string_utilities (program), 1302

test_tables (program), 1302

test_vectors (program), 1302

test_zhao2009_concentration (program), 1286

test_zhao2009_flat (program), 1284

test_zhao2009_open (program), 1286

testfuncdouble0 (function), 1296

tests.abundances.F90 (file), 1286

tests.arrays.F90 (file), 1287

tests.black_hole_fundamentals.F90 (file), 1287

tests.bug745815.F90 (file), 1287

tests.comoving_distance.dark_energy.F90 (file), 1287

tests.comoving_distance.EdS.F90 (file), 1287

tests.comoving_distance.open.F90 (file), 1288

tests.comparisons.F90 (file), 1288

tests.cosmic_age.cosmological_constant.F90 (file), 1288

tests.cosmic_age.dark_energy.closed.F90 (file), 1289

tests.cosmic_age.dark_energy.cosmological_constant.F90 (file), 1289

tests.cosmic_age.dark_energy.omegaMinusOneThird.F90 (file), 1289

tests.cosmic_age.EdS.F90 (file), 1288

tests.cosmic_age.open.F90 (file), 1291

tests.geometry.coordinate_systems.F90 (file), 1291

tests.halo_mass_function.Tinker.F90 (file), 1291

tests.hashes.cryptographic.F90 (file), 1292

tests.hashes.F90 (file), 1291

tests.hashes.perfect.F90 (file), 1292

tests.integration.F90 (file), 1292

tests.integration.functions.F90 (file), 1292

tests.interpolation.2D.F90 (file), 1293

tests.interpolation.F90 (file), 1293

tests.IO.HDF5.F90 (file), 1282

tests.IO.XML.F90 (file), 1283

tests.kepler_orbits.F90 (file), 1293

tests.linear_growth.cosmological_constant.F90 (file), 1294

tests.linear_growth.dark_energy.F90 (file), 1294

tests.linear_growth.EdS.F90 (file), 1294

tests.linear_growth.open.F90 (file), 1294
tests.make_ranges.F90 (file), 1295
tests.mass_distributions.F90 (file), 1295
tests.math_special_functions.F90 (file), 1295
tests.meshes.F90 (file), 1295
tests.NFW96_concentration.dark_energy.F90 (file), 1283
tests.nodes.F90 (file), 1296
tests.nodes.task.F90 (file), 1296
tests.ODE_solver.F90 (file), 1283
tests.ODE_solver.functions.F90 (file), 1283
tests.power_spectrum.F90 (file), 1296
tests.Prada2011_concentration.F90 (file), 1284
tests.root_finding.F90 (file), 1297
tests.root_finding.functions.F90 (file), 1297
tests.search.F90 (file), 1297
tests.sigma.F90 (file), 1298
tests.sort.F90 (file), 1298

tests.spherical_collapse.dark_energy.constantEoSminus0.6.F90 (file), 1298

tests.spherical_collapse.dark_energy.constantEoSminus0.8.F90 (file), 1300

tests.spherical_collapse.dark_energy.constantEoSminusHalf.F90 (file), 1300

tests.spherical_collapse.dark_energy.constantEoSminusTwoThirds.F90 (file), 1300

tests.spherical_collapse.dark_energy.EdS.F90 (file), 1298

tests.spherical_collapse.dark_energy.lambda.F90 (file), 1301

tests.spherical_collapse.dark_energy.open.F90 (file), 1301

tests.spherical_collapse.flat.F90 (file), 1301

tests.spherical_collapse.open.F90 (file), 1301

tests.string_utilities.F90 (file), 1302

tests.tables.F90 (file), 1302

tests.tree_branch_destroy.F90 (file), 1302

tests.vectors.F90 (file), 1302

tests.Zhao2009_algorithms.dark_energy.F90 (file), 1286

tests.Zhao2009_algorithms.EdS.F90 (file), 1284

tests.Zhao2009_algorithms.open.F90 (file), 1286

tests_bug745815 (program), 1287

tests_comoving_distance_dark_energy (program), 1288

tests_comoving_distance_eds (program), 1287

tests_comoving_distance_open (program), 1288

tests_cosmic_age_cosmological_constant (program), 1289

tests_cosmic_age_dark_energy_cold (program), 1289

tests_cosmic_age_dark_energy_cosmological_constant (program), 1289

tests_cosmic_age_dark_energy_omega_minus_one_third (program), 1289

tests_cosmic_age_eds (program), 1288

tests_excursion_sets (program), 599

tests_halo_mass_function_tinker (program), 1291

tests_io_hdf5 (program), 1282

tests_io_xml (program), 1283

tests_kepler_orbits (program), 1293

tests_linear_growth_cosmological_constant (program), 1294

tests_linear_growth_dark_energy (program), 1294
tests_linear_growth_eds (program), 1294
tests_linear_growth_open (program), 1294
tests_power_spectrum (program), 1296
tests_sigma (program), 1298
tests_sphericalCollapseDarkEnergyEds (program), 1298
tests_sphericalCollapseDarkEnergyLambda (program), 1301
tests_sphericalCollapseDarkEnergyOmegaHalf (program), 1300
tests_sphericalCollapseDarkEnergyOmegaTwoThirds (program), 1300
tests_sphericalCollapseDarkEnergyOmegaZeroPointEight (program), 1300
tests_sphericalCollapseDarkEnergyOmegaZeroPointSix (program), 1298
tests_sphericalCollapseDarkEnergyOpen (program), 1301
tests_sphericalCollapseFlat (program), 1301
tests_sphericalCollapseOpen (program), 1302
testvoidfunc (subroutine), 1296
thermodynamics.ideal_gases.F90 (file), 1303
thermodynamics.radiation.F90 (file), 1303
thermodynamics_radiation (module), 1303
tidal_radius_root (function), 623
time_from_comoving_distance (function), 676
time_from_comoving_distance_matter_dark_energy (function), 679
time_from_comoving_distance_matter_lambda (function), 683
time_of_collapse (function), 1240
time_step_get (function), 850
timesteprange (type), 773
transfer_function (function), 1277
transfer_function_bbks (module), 1275
transfer_function_bbks_initialize (subroutine), 1275
transfer_function_bbks_make (subroutine), 1275
transfer_function_bbks_state_retrieve (subroutine), 1275
transfer_function.bbks_state_store (subroutine), 1275
transfer_function_cmbfast (module), 1275
transfer_function_cmbfast_initialize (subroutine), 1276
transfer_function_cmbfast_label (function), 1337
transfer_function_cmbfast_make (subroutine), 1276
transfer_function_eisenstein_hu (module), 1276
transfer_function_eisenstein_hu_initialize (subroutine), 1276
transfer_function_eisenstein_hu_make (subroutine), 1276
transfer_function.eisenstein_hu_state_retrieve (subroutine), 1276
transfer_function.eisenstein_hu_state_store (subroutine), 1277
transfer_function_file_initialize (subroutine), 1278
transfer_function_file_read (subroutine), 1278
transfer_function_initialize (subroutine), 1277
transfer_function_logarithmic_derivative (function), 1277
transfer_function_named_file_format_version (function), 1278
transfer_function_named_file_read (subroutine), 1278
transfer_function_null (module), 1278
transfer_function_null_initialize (subroutine), 1279
transfer_function_null_make (subroutine), 1279
transfer_function_state_retrieve (subroutine), 1277
transfer_function_tabulate_template (interface), 1277
transfer_function_tabulate_template (subroutine), 1278
transfer_functions (module), 1277
transfer_functions_file (module), 1278
tree_branch_mass (function), 827
tree_branch_mass_template (function), 828
tree_branch_mass_template (interface), 827
tree_branching_initialize (subroutine), 828
tree_branching_modifiers_initialize (subroutine), 833
tree_branching_probability (function), 828
tree_branching_probability_template (function), 828
tree_branching_probability_template (interface), 828
tree_maximum_step (function), 828
tree_maximum_step_template (function), 828
tree_maximum_step_template (interface), 828
tree_node_component_builder (subroutine), 1109
tree_node_compute_derivatives (subroutine), 848
tree_node_constructor (function), 1109
tree_node_copy_node_to (subroutine), 1109
tree_node_create_event (function), 1109
tree_node_evolve (subroutine), 848
tree_node_evolve_error_analyzer (subroutine), 848
tree_node_evolve_initialize (subroutine), 849
tree_node_get_earliest_progenitor (function), 1109
tree_node_get_last_satellite (function), 1109
tree_node_index (function), 1109
tree_node_index_set (subroutine), 1109
tree_node_is_accurate (function), 849
tree_node_is_on_main_branch (function), 1110
tree_node_is_primary_progenitor (function), 1110
tree_node_is_primary_progenitor_of_index (function), 1110
tree_node_is_primary_progenitor_of_node (function), 1110
tree_node_is_satellite (function), 1110
tree_node_merges_with_node (function), 1110
tree_node_move_components (subroutine), 1110
tree_node_ode_step_rates_initialize (subroutine), 1110
tree_node_ode_step_scales_initialize (subroutine), 1110
tree_node_odes (function), 849
tree_node_odes_error_handler (subroutine), 849
tree_node_promote (subroutine), 849
tree_node_remove_from_host (subroutine), 1110
tree_node_remove_from_mergee (subroutine), 1111
tree_node_removepaired_event (subroutine), 1111
tree_node_type (function), 1111
tree_node_unique_id (function), 1111
tree_node_unique_id_set (subroutine), 1111
tree_node_walk_branch (subroutine), 1111
tree_node_walk_branch_with_satellites (subroutine), 1111
tree_node_walk_tree (subroutine), 1112
tree_node_walk_tree_under_construction (subroutine), 1113
tree_node_walk_tree_with_satellites (subroutine), 1113
tree_subresolution_fraction (function), 828
tree_subresolution_fraction_template (function), 829
tree_subresolution_fraction_template (interface), 828
treemetadata (type), 913
treenode (type), 1113
treenodedestroy (subroutine), 1113
treenodeinitialize (subroutine), 1113
treenodelist (type), 1113
triangular_shaped_cloud_integral (function), 876
trim (interface), 822
trim_ (function), 822

unit_tests (module), 1358
unit_tests_begin_group (subroutine), 1362
unit_tests_end_group (subroutine), 1362
unit_tests_finish (subroutine), 1362
unitsmetadata (type), 913
utility.arrays.F90 (file), 1317
utility.command_arguments.F90 (file), 1319
utility.date_and_time.F90 (file), 1320
utility.files.F90 (file), 1320
utility.hashes.cryptographic.F90 (file), 1323
utility.hashes.F90 (file), 1321
utility.hashes.perfect.F90 (file), 1324
utility.input_parameters.F90 (file), 1325
utility.IO.HDF5.F90 (file), 1303
utility.IO.XML.F90 (file), 1315
utility.kind_numbers.F90 (file), 1337
utility.memory_management.F90 (file), 1339
utility.memory_usage.cpp (file), 1353
utility.string_handling.F90 (file), 1353
utility.unit_tests.F90 (file), 1358

validate_isolated_halos (subroutine), 844
value_integer_scalar_ch (function), 1323
value_integer_scalar_i (function), 1323
value_integer_scalar_vs (function), 1323
values_agree (interface), 863
values_agree_double (function), 863
values_agree_real (function), 864
values_differ (interface), 864
values_differ_double (function), 864
values_differ_real (function), 864
values_integer_scalar (subroutine), 1323
var_str (interface), 822
var_str_ (function), 822
variance_integral (function), 1238
variance_integrand (function), 1238
variance_integrand_tophat (function), 1238
varying_string (type), 822
vector_magnitude (function), 827
vector_product (function), 827
vectors (module), 827
velocity_dispersion_integrand (function), 734
verify (interface), 822
verify_ch_vs (function), 822
verify_vs_ch (function), 823
verify_vs_vs (function), 823
virial_densities_bryan_norman (module), 1279
virial_densities_fixed (module), 1281
virial_densities_kitayama_suto1996 (module), 1281
virial_density_bryan_norman (subroutine), 1279
virial_density_bryan_norman_initialize (subroutine), 1279
virial_density_contrast (module), 1279
virial_density_contrast_initialize (subroutine), 1280
virial_density_contrast_retabulate (subroutine), 1280
virial_density_contrast_state_retrieve (subroutine), 1280
virial_density_fixed (subroutine), 1281
virial_density_fixed_initialize (subroutine), 1281
virial_density_kitayama_suto1996 (subroutine), 1281
virial_density_kitayama_suto1996_initialize (subroutine), 1281
virial_orbital_parameters (function), 1178
virial_orbital_parameters_benson2005 (function), 1177
virial_orbital_parameters_benson2005_initialize (subroutine), 1177
virial_orbital_parameters_benson2005_snapshot (subroutine), 1177
virial_orbital_parameters_benson2005_state_retrieve (subroutine), 1177
virial_orbital_parameters_benson2005_state_store (subroutine), 1177
virial_orbital_parameters_benson2005_fixed (function), 1177
virial_orbital_parameters_fixed (subroutine), 1177
virial_orbital_parameters_fixed_initialize (subroutine), 1179
virial_orbital_parameters_wetzel2010 (function), 1178
virial_orbital_parameters_wetzel2010_initialize (subroutine), 1178
virial_orbital_parameters_wetzel2010_snapshot (subroutine), 1179
virial_orbital_parameters_wetzel2010_state_retrieve (subroutine), 1179
virial_orbital_parameters_wetzel2010_state_store (subroutine), 1179
virial_orbital_parameters_benson2005 (module), 1176
virial_orbits (module), 1177
virial_orbits_benson2005 (module), 1177
virial_orbits_fixed (module), 1179
virial_orbits_wetzel2010 (module), 1178
wind_energy_integrand (function), 1219
write_parameter (subroutine), 1337

xermsg (subroutine), 601
xi_integrand (function), 840
xml_array_length (function), 1315
xml_array_read (interface), 1315
xml_array_read_one_column (subroutine), 1315
CODE INDEX

xml_array_read_static (interface), 1316
xml_array_read_static_one_column (subroutine), 1316
xml_array_read_two_column (subroutine), 1316
xml_extrapolation_element_decode (subroutine), 1316
xml_get_first_element_by_tag_name (function), 1316
xml_list_array_read_one_column (subroutine), 1316
xml_list_character_array_read_static_one_column (subroutine), 1316
xml_list_double_array_read_static_one_column (subroutine), 1316
xml_list_integer_array_read_static_one_column (subroutine), 1317
xml_path_exists (function), 1317
xray_absorption_ism_wilms2000 (program), 601
XRay_Absorption_ISM_Wilms2000.F90 (file), 601
xwrite (subroutine), 603