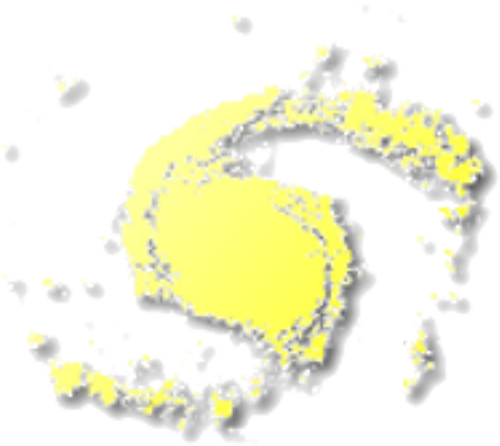
Galacticus



A semi-analytic galaxy formation code.
Analyzing Galacticus Outputs Using Perl.

Contents

1 About Galacticus

Galacticus is a semi-analytic model of galaxy formation. This document describes a collection of modules implemented in Perl designed for interacting with and analyzing the output of Galacticus models.

1.0.1 License

2 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.

2.1 General Structure of Output File

Figure 2.1 shows the structure of a typical Galacticus output file. The various groups and subgroups are described below.

2.1.1 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 §2.2), the UUID can be loaded into the data structure using:

```
&HDF5::Get_UUID($model);
```

The UUID is then available as `$model->{'uuid'}`.

2.1.2 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;

```
outputFile.hdf5
 |
 +-> UUID                                  Attribute {1}
 |
 +-> Build                                 Group
 |    |
 |    +-> FGSL_library_version             Attribute {1}
 |    +-> FoX_library_version              Attribute {1}
 |    +-> GSL_library_version              Attribute {1}
 |    +-> HDF5_library_version             Attribute {1}
 |    +-> make_CCOMPILER                   Attribute {1}
 |    +-> make_CCOMPILER_VERSION           Attribute {1}
 |    +-> make_CFLAGS                      Attribute {1}
 |    +-> make_CPPCOMPILER                 Attribute {1}
 |    +-> make_CPPCOMPILER_VERSION         Attribute {1}
 |    +-> make_CPPFLAGS                    Attribute {1}
 |    +-> make_FCCOMPILER                  Attribute {1}
 |    +-> make_FCCOMPILER_VERSION          Attribute {1}
 |    +-> make_FCFLAGS                     Attribute {1}
 |    +-> make_FCFLAGS_NOOPT               Attribute {1}
 |    +-> make_MODULETYPE                  Attribute {1}
 |    +-> make_PREPROCESSOR                Attribute {1}
 |    +-> sourceChangeSetDiff              Dataset   {1}
 |    +-> sourceChangeSetMerge             Dataset   {1}
 |
 +-> Outputs                               Group
 |    |
 |    +-> Output1                          Group
 |    |    |
 |    |    +-> nodeData                    Group
 |    |    |     |
 |    |    |     +-> nodeProperty1         Dataset {<nodeCount>}
 |    |    |     +-> ...                   Dataset {<nodeCount>}
 |    |    |     +-> ...                   Dataset {<nodeCount>}
 |    |    |     +-> ...                   Dataset {<nodeCount>}
 |    |    |     +-> nodePropertyN         Dataset {<nodeCount>}
 |    |    |
 |    |    +-> mergerTreeCount             Dataset {<treeCount>}
 |    |    |
 |    |    +-> mergerTreeIndex             Dataset {<treeCount>}
 |    |    |
 |    |    +-> mergerTreeStartIndex        Dataset {<treeCount>}
 |    |    |
 |    |    +-> mergerTreeWeight            Dataset {<treeCount>}
 |    |    |
 |    |    +-> mergerTree1                 Group             [optional]
 |    |    |     |
 |    |    |     +-> nodeProperty1         Reference
 |    |    |     +-> ...                   Reference
 |    |    |     +-> ...                   Reference
 |    |    |     +-> ...                   Reference
 |    |    |     +-> nodePropertyN         Reference
 |    |    |
 |    |    x-> ...                         Group             [optional]
 |    |    x-> ...                         Group             [optional]
 |    |    x-> ...                         Group             [optional]
 |    |    x-> mergerTree<treeCount>       Group             [optional]
 |    |    |
```

`make_CFLAGS` The flags passed to the C compiler;

`make_CPPCOMPILER` The C++ compiler command used;

`make_CPPCOMPILER_VERSION` The C++ compiler version information;

`make_CPPFLAGS` The flags passed to the C++ compiler;

`make_FCCOMPILER` The Fortran compiler command used;

`make_FCCOMPILER_VERSION` The Fortran compiler version information;

`make_FCFLAGS` The flags passed to the Fortran compiler;

`make_FCFLAGS_NOOPT` The flags passed to the Fortran compiler for unoptimized compiles;

`make_MODULETYPE` The Fortran module type identifier string;

`make_PREPROCESSOR` The preprocessor command used.

Additionally, two datasets are included which store details of the Galacticus source changeset. `sourceChangeSetMerge` contains the output of "`hg bundle -t none`", that is, it contains a Mercurial changegroup that incorporates any changes made to the current branch relative to the main Galacticus branch. `sourceChangeSetDiff` contains the output of "`hg diff`", that is, all differences between the source code in the working directory and that which has been committed to Mercurial. Used together, these two datasets allow the precise source code used to run the model to be recovered from the main branch Galacticus source.

2.1.3 Filters

For each broadband filter used in the Galacticus model run an entry is added to the datasets in this group. Currently, two datasets are generated:

`name` The name of each filter used.

`wavelengthEffective` The effective wavelength, $\lambda_{\rm eff}$ (defined as $\lambda_{\rm eff} = \int_0^\infty \lambda R(\lambda) {\rm d}\lambda \big/ \int_0^\infty R(\lambda) {\rm d}\lambda$, where $R(\lambda)$ is the filter response) of the filter in Å.

2.1.4 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. If a parameter has subparameters, a group is created having the same name as the parameter, which will contain attributes corresponding to each subparameter. 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.

2.1.5 Version

The `Version` group contains a record of the Galacticus version used for this model, storing the major and minor version numbers, the revision number and the Mercurial revision and hash (if the code is being maintained using Mercurial, otherwise a value of -1 is entered or the revision and the hash attribute is empty). Additionally, the time at which the model was run is stored and, if the `galacticusConfig.xml` file (see §3.1) is present and contains contact details, the name and e-mail address of the person who ran the model.

2.1.6 globalHistory

The `globalHistory` group stores volume averaged properties of the model universe as a function of time. Currently, the properties stored are:

`historyTime` Cosmic time (in Gyr);

`historyExpansion` Expansion factor;

`historyStarFormationRate` Volume averaged star formation rate (in $M_\odot$/Gyr/Mpc^3).

`historyDiskStarFormationRate` Volume averaged star formation rate in disks (in $M_\odot$/Gyr/Mpc^3).

`historySpheroidStarFormationRate` Volume averaged star formation rate in spheroids (in $M_\odot$/Gyr/Mpc^3).

`historyStellarDensity` Volume averaged stellar mass density (in $M_\odot$/Mpc^3).

`historyDiskStellarDensity` Volume averaged stellar mass density in disks (in $M_\odot$/Mpc^3).

`historySpheroidStellarDensity` Volume averaged stellar mass density in spheroids (in $M_\odot$/Mpc^3).

`historyGasDensity` Volume averaged cooled gas density (in $M_\odot$/Mpc^3).

`historyNodeDensity` Volume averaged resolved node density (in $M_\odot$/Mpc^3).

Dimensionful datasets have a `unitsInSI` attribute which gives their units in the SI system.

2.1.7 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.

Output is controlled by parameters given within the `mergerTreeOutput` section of the parameter file. Current options are:

`outputMergerTrees` If `true` then each merger tree is output to the relevant sub-group at each output time (see §2.1.7). Otherwise merger trees are not output. [Default: `true`.]

`outputReferences` If `true` then an HDF5 reference dataset is written for each merger tree subgroup (see §2.1.7). [Default: `false`.]

`galacticFilterMethod` A "galactic filter" (see §17.4.1) which is applied to each node in the tree to determine whether or not it should be output. By combinding multiple filters it is possible to construct arbitrarily complex criteria for output. [Default: `always`.]

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 §2.1.7) which gives the weight (in Mpc^{-3}) which should be assigned to this tree (and all nodes in it) to create a volume-averaged sample (see §2.3.1). 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 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.)

2.1.8 Optional Outputs

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

Redshifts

The redshift corresponding to the time at which a node was last isolated can be output by setting `[outputNodeRedshifts]` to `true`. This quantity will be output as `basicRedshiftLastIsolated`.

Mass Accretion Histories

A mass accretion history (i.e. mass as a function of time) for the main branch in each merger tree can be output by setting `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;

`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.

[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.

Merger Tree Dump

A full dump of merger tree structure by setting `mergerTreeStructureDump=true`. In this case, files will be dumped to the directory specified by `[mergerTreeStructureDumpDirectory]` for each merger tree with final mass between `[mergerTreeStructureDumpMassMinimum]` and `[mergerTreeStructureDumpMassMaximum]`. Each tree is dumped to a file named "`mergerTreeDump:<treeIndex>:1.gv`" in the specified directory in GraphViz format.

Conditional Mass Functions

Setting `[mergerTreeComputeConditionalMassFunction]=true` will cause conditional mass functions to be computed and output to the Galacticus output file in a group named "`conditionalMassFunction`". The mass functions are binned in parent halo mass, and the mass ratio of the progenitor to parent halo. Bins are logarithmically spaced in mass (and mass ratio), with the range and number of bins controlled by the parameters:

- `[mergerTreeComputeConditionalMassFunctionParentMassCount]`;
- `[mergerTreeComputeConditionalMassFunctionParentMassMinimum]`;
- `[mergerTreeComputeConditionalMassFunctionParentMassMaximum]`;
- `[mergerTreeComputeConditionalMassFunctionMassRatioCount]`;
- `[mergerTreeComputeConditionalMassFunctionMassRatioMinimum]`;
- `[mergerTreeComputeConditionalMassFunctionMassRatioMaximum]`.

The resulting parent masses and mass ratios are written to datasets `massParent` and `massRatio` respectively. Parent and progenitor halos are defined at a set of redshifts defined by the arrays `[mergerTreeComputeConditionalMas` and `[mergerTreeComputeConditionalMassFunctionProgenitorRedshifts]`, which are written to datasets `redshiftParent` and `redshiftProgenitor`. The resulting conditional masses functions are written to datasets `conditionalMassFunction` and `conditionalMassFunctionError`.

In addition to standard progenitor mass functions, the progenitor mass function conditioned on progenitor rank (i.e. $1^{\rm st}$ most massive, $2^{\rm nd}$, $\ldots$, $n^{\rm th}$ most massive progenitor) is computed and output to the datasets `primaryProgenitorMassFunction` and `primaryProgenitorMassFunctionError`. The depth (i.e. n) is specifed by `[mergerTreeComputeConditionalMassFunctionPrimaryProgenitorDepth]`.

Finally, the progenitor mass functoin conditioned on recent formation is computed and output to the datasets `formationRateFunction` and `formationRateFunctionError`. To be considered "recently formed" a progenitor must have formed between t and $t(1-\Delta)$ where t is the progenitor time and $\Delta=$`[mergerTreeConditionalMassFunctionFormationRateTimeFraction]`.

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;

`childIndex` The index of this node's first child node;

`parentIndex` The index of this node's parent node;

`siblingIndex` 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);

ark 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);

tree final descendent If `outputFinalDescendentIndices=true` then an additional dataset is included:

`finalDescendentIndex` The index of the final descendent that this node will reach in its merger trees;

2.2 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:

```
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'}."\n";
```

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.

2.2.1 Derived Properties

Derived properties can be created by giving defining functions along with a regular expression string that allows them to be matched. For example, the `Galacticus::Baryons` module implements a hot gas fraction property called `hotHaloFraction` or `hotHaloFrac`. It has the following form:

```
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 `%HDF5::galacticusFunctions` hash. The key gives a regular expression which matches to the name of the property to be defined. The value of the key gives a reference to a subroutine to be called to evaluate this expression. The subroutine is defined below. When called, it receives the `$model` structure along with the name of the requested property. The subroutine should then simply evaluate the requested property and store it in the appropriate location within `$model`. Note that the subroutine can request additional datasets be loaded (as happens above where `hotHaloMass` and `nodeMass` are requested) if they are needed for its calculations.

Available Derived Properties

`mergerTreeIndex` The index of the merger tree in which the galaxy is found. Provided by: `Galacticus::HDF5`.

`redshift` The redshift at which the galaxy exists. Provided by: `Galacticus::Time`.

`time` The cosmic time (in Gyr) at which the galaxy exists. Provided by: `Galacticus::Time`.

`expansionFactor` The expansion factor at which the galaxy exists. Provided by: `Galacticus::Time`.

`stellarMass` The sum of disk and spheroid stellar masses. Provided by: `Galacticus::StellarMass`.

`massColdGas` The sum of disk and spheroid cold gas masses. Provided by: `Galacticus::GasMass`.

`starFormationRate` The sum of disk and spheroid star formation rates. Provided by: `Galacticus::StellarMass`.

`hostNodeMass` For isolated nodes, the node mass. For non-isolated nodes, the mass of the isolated node in which the node resides. Provided by: `Galacticus::HostNode`.

`stellarMass` The sum of disk and speheroid stellar masses (or, whichever of these exist in the model). Provided by: `Galacticus::StellarMass`.

`hotHalo(Fraction|Frac)` The fraction the node's mass in the hot gas halo. Provided by: `Galacticus::Baryons`.

`inclination` A randomly selected inclination for the disk (in degrees). Provided by: `Galacticus::Inclination`.

`^(disk|bulge)StellarLuminosity:.*:dustAtlas(\[faceOn\])$` Dust-extingiushed luminosities for disk and bulge found by interpolating in the dust tables of Ferrara et al. [1999]. If the `[faceOn]` qualifier is present, extinctions are computed assuming that the disk is observed face-on, otherwise a random inclination is used. Optionally, the dust atlas file to used can be specified via `$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). *Note:* Dust attenuated luminosities for all galaxies, all filters, and all output redshifts can be computed and stored back to the hierarchical data format (HDF5) file using a simple script as described in §4.4. Provided by: `Galacticus::DustAttenuation`.

`^(disk|bulge)LuminositiesStellar:.*:dustCharlotFall2000$` Dust-extingiushed luminosities for disk and bulge found using the model of Charlot and Fall [2000]. Provided by: `Galacticus::DustCharlotFall2000`.

`^totalStellarLuminosity:.*:dustAtlas(\[faceOn\])$` (Optionally dust-extingiushed) 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-extingiushed) 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 §2.3.1 for a full description).

`luminosityDistance` The luminosity distance (in Mpc) to the galaxy—provided by `Galacticus::Survey` (see §2.3.1 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 §2.3.1 for a full description).

`redshift` The redshift at which the galaxy is observed—provided by `Galacticus::Survey` (see §2.3.1 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 §2.3.1 for a full description).

`angularDiameterDistance` The angular diameterer distance (in Mpc) to the galaxy—provided by `Galacticus::Survey` (see §2.3.1 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 §2.3.1 for a full description).

`^grasilFlux[\d\.]+microns` The flux at the given wavelength (specific in microns) of the galaxy as computed by the `Grasil` code (see §2.5 for a full description).

`^grasilInfraredLuminosity` The total infrared (8–1000μm) luminosity of the galaxy as computed by the `Grasil` code (see §2.5 for a full description).

`^grasilFlux:([^:]+)` The flux (in Janskys) of the galaxy as computed by the `Grasil` code integrated under the specified filter (see §2.5 for a full description).

`^luminosity:grasil:([^:]+):([^:]+)` The luminosity (in units of the zero point of the AB magnitude system) of the galaxy as computed by the `Grasil` code integrated under the specified filter and in the specified frame (see §2.5 for a full description).

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

$$\frac{S_{850\mu\mathrm{m}}}{\mathrm{Jy}} = A \left(\frac{\dot{M}_\star}{100 M_\odot \mathrm{Gyr}^{-1}} \right)^\alpha \left(\frac{R_\mathrm{dust} M_\mathrm{metals,gas}}{10^8 M_\odot} \right)^\beta, \tag{2.1}$$

where R_dust is the dust-to-metals ratio, $\dot{M}_\star$ is the total star formation rate in the galaxy and $M_\mathrm{metals,gas}$ 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 galaxcy at $z \approx 2$. The parameters of the fit can be specified by setting elements of `$model->{'haywardSubMmFit'}`: `{'dustToMetalsRatio'}`$\equiv R_\mathrm{dust}$, `{'fitNormalization'}`$\equiv A$, `{'starFormationRateExponent'}`$\equiv \alpha$, and `{'dustMassExponent'}`$\equiv \beta$. If these elements are not present the default values of $A = 0.65 \times 10^{-3}$, $R_\mathrm{dust} = 0.61$, $\alpha = 0.42$ and $\beta = 0.58$ Hayward et al. [2010] will be used instead. Provided by: `Galacticus::SubMmFluxesHayward`.

`^(disk|spheroid|total)LymanContinuumLuminosity:z[\d\.]+$` The luminosity (in units of 10^{50}photons/s) of the Lyman continuum radiation of the disk or spheroid component, or total of the two at the specified redshift. The rest-frame `Lyc` filter must have been computed in Galacticus to allow this luminosity to be computed. If the `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`.

`^agnLuminosity:[^:]+:[^:]+:z[\d\.]+(:alpha[0-9\-\+\.]+)??$` 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 `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`.

`^columnDensity(disk|Spheroid)??$` The column density of hydrogen (in units of cm^{-2}) along the line of sight to the center of the galaxy. If a component is specified the calculation is performed for that component, otherwise the sum of disk and spheroid column densities is computed. Provided by: `Galacticus::ColumnDensity`. For the exponential disk, the column density is given by:

$$N_\mathrm{H} = \frac{X_\mathrm{H}}{m_\mathrm{H}} \frac{M_\mathrm{ISM}}{4\pi h r_\mathrm{d}^3} \int_0^\infty \exp(-r/r_\mathrm{d}) \mathrm{sech}^2(z/hr_\mathrm{d}) \mathrm{d}l, \tag{2.2}$$

where r_d is the disk scale length, h is the ratio of vertical scale height to radial scale length, X_H is the mass fraction in hydrogen, m_H is the mass of the hydrogen atom and l is distance along the line of sight. Writing $z = r/\tan i$ for a disk at inclination i, and $l = \sqrt{r^2+z^2} = r\sqrt{1+1/\tan^2 i}$ this becomes:

$$N_\mathrm{H} = \frac{X_\mathrm{H}}{m_\mathrm{H}} \frac{M_\mathrm{ISM}}{4\pi h r_\mathrm{d}^2} \sqrt{1+1/\tan^2 i} \int_0^\infty \exp(-x) \mathrm{sech}^2(x/h\tan i) \mathrm{d}x. \tag{2.3}$$

The integral can be evaluated to give:

$$N_\mathrm{H} = \frac{X_\mathrm{H}}{m_\mathrm{H}} \frac{M_\mathrm{ISM}}{8\pi h r_\mathrm{d}^2} \sqrt{1+1/\tan^2 i} H \left\{ H \left[\psi\left(-\frac{H}{4}\right) - \psi\left(\frac{1}{2}-\frac{H}{4}\right)\right] - 2 \right\}, \tag{2.4}$$

where $H = h\tan i$ and $\psi(x)$ is the digamma function.

`^peakSFR$` The peak star formation rate for each galaxy, measured from the `starFormationHistories` output group (see §17.4.3). The peak star formation rate reported is therefore that when averaged over the bins used by the star formation history output method (see §17.4.3). Provided by: `Galacticus::SFH`.

`^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: `Galacticus::LensingAmplification`.

`^(disk|spheroid|total)LineLuminosity:[^:]+(:[^:]+){0,2}:z[\d\.]+$` Returns the luminosity of the named emission line, for the named component, at the given redshift in units of Solar luminosities. Optinally, a filter may be provided in which case the emission line luminosity under that filter is returned in units of AB maggies. For example, `totalLineLuminosity:balmerAlpha6563:rest:z0.0000` returns the luminosity of the Hα line at $z = 0$. See §4.6.3 for more details. Provided by: `Galacticus::EmissionLines`.

2.2.2 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 `Galacticus::HaloModel` Perl module. To use this module, Galacticus must be run with `[outputHaloModelData]=true` (see §14.8) to output data on halo profiles and power spectra. To perform halo model calculations, simply use this module in a Perl script, initialize the data hash, `%dataHash`, used for the `Galacticus::HDF5` module, and construct a PDL, `$selectedGalaxies`, which contains the indices (not the node indices, but the positions within the PDL arrays read in from the Galacticus output file) of galaxies for which the clustering is to be computed. A power spectrum can then be computed using:

```
($waveNumber,$linearPowerSpectrum,$galaxyPowerSpectrum)
    = &HaloModel::Compute_Power_Spectrum($model,$selectedGalaxies,space => "redshift");
```

The PDLs returned contain a list of comoving wavenumbers, the linear power spectrum of matter at the selected time and the (non-linear) power spectrum of the selected galaxies. If the `space` option is set to `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:

```
($separations,$galaxyCorrelationFunction)
    = &HaloModel::Compute_Correlation_Function($waveNumber,$galaxyPowerSpectrum
        ,$separationMinimum,$separationMaximum,$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 separtion (in Mpc) and correlation function corresponding to the input power spectrum.

2.3 Topics in Analysis of Galacticus Outputs

2.3.1 Building Volume Limited Samples

The `mergerTreeWeight` property (see §2.1.7) property specifies the weight to be assigned to each merger tree in a model to construct a representative (i.e. volume limited) sample of galaxies. Galacticus does not typically generate every merger tree in a fixed volume of the Universe (as an N-body simulation might for example) as it's generally a waste of time to simulate millions of low mass halos and only a small number of high mass halos. The `mergerTreeWeight` factors correct for this sampling. If merger trees are being built, then the `mergerTreeWeight`, w_i, for each tree of mass M_i (where the trees are ranked in order of increasing mass) is given by

$$w_i = \int_{M_{\rm min}}^{M_{\rm max}} n(M) {\rm d}M, \tag{2.5}$$

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

$$M_{\rm min} = \sqrt{M_{i-1} M_i}, \tag{2.6}$$

$$M_{\rm min} = \sqrt{M_i M_{i+1}}. \tag{2.7}$$

Suppose, for example, that we wish to construct a luminosity function of galaxies. In particular, we consider a luminosity bin k which extends from $L_k - \Delta k/2$ to $L_k + \Delta k/2$. If tree i contains N_i galaxies

with luminosities $l_{i,j}$, where j runs from 1 to N_i, then the luminosity function in this bin is given by:

$$\phi_k = \sum_i \sum_{j=1}^{N_i} \begin{cases} w_i & \text{if } L_k - \Delta k/2 < l_{i,j} \leq L_k + \Delta k/2 \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

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 §14.9). 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 preceeding and succeeding outputs (for the latest output the range is extended to $z = 0$, while for the earliest output the range is truncated at the redshift of the output itself).

Within this redshift range, galaxies are assigned a comoving distance (property `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 a `angularWeight` property. This gives the mean number of each galaxy that would be found in a solid angle of one steradian.

2.4 Postprocessing Scripts

2.5 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 §2.2.1. For a more detailed treatment of dust extinction, and the re-emission of starlight by dust, Galacticus is able to interface with the Grasil radiative transfer code described by Silva et al. [1998].

To process a Galacticus galaxy through Grasil use the following method:

1. Run Galacticus to generate galaxies. Grasil requires a detailed star formation history for each galaxy it processes. Therefore, you should set `[starFormationHistoriesMethod]=metallicity split` in your input parameter file. Other parameters controlling the details of the star formation history recording are discussed in §17.4.3. Note that you should ensure that the history is recorded with sufficient precision to permit an accurate calculation by Grasil. Additionally, you may want to consider using the same stellar population data in Galacticus as is used by Grasil—suitable files in Galacticus format can be downloaded from the Galacticus web site.

2. Select a galaxy from the output to process through Grasil. You will need to know the output number, tree index and node index of the galaxy;

3. Run the `Extract_Star_Formation_History_for_Grasil.pl` script to extract the star formation history for this galaxy in a format suitable for input into Grasil:

```
scripts/aux/Extract_Star_Formation_History_for_Grasil.pl <inputFile> <outputIndex> \
   <treeIndex> <nodeIndex> <grasilFile> [<plotFile>]
```

where `inputFile` is the name of the GALACTICUS model file, `outputIndex`, `treeIndex` and `nodeIndex` are the quantities described above that identify the galaxy of interest and `grasilFile` is the name of the file to which the star formation history should be written. GRASIL convention dictates that this file should have the suffix `.dat`. The optional `plotFile` is the name of a file to which a plot of the star formation history will be written.

4. Create a suitable input parameter file for `Grasil`, with the same name as your star formation file created above, but with the suffix `.par`. An example of such a file is given in `aux/Grasil/grasilExample.par` —refer to the GRASIL documentation for details of the parameters and how to control GRASIL.

5. Download the GRASIL executable from here and supporting data files from here and unpack them[2].

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.

2.5.1 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 §2.2.1 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>`μm.

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 (8–1000μm) luminosity of galaxies. Note that these properties require that the `Galacticus::Survey` module be used to provide redshifts for galaxies (see §2.3.1).

The module will automatically run GRASIL using the parameters given in `data/grasilBaseParameters.txt`, subject the modifications specified in the `grasilOptions` element of `$model`. Allowed options are:

`$dataSet->{'grasilOptions'}->{'dustToMetalsRatio}` Sets the dust to metals ratio used in GRASIL;

`$dataSet->{'grasilOptions'}->{'includePAHs'}` Set to 0/1 to switch off/on calculations of PAH features in GRASIL;

`$dataSet->{'grasilOptions'}->{'fluctuatingTemperatures'}` Set to 0/1 to switch off/on calculations of fluctuating dust grain temperatures in GRASIL;

`$dataSet->{'grasilOptions'}->{'wavelengthCount'}` Specifies the number of wavelengths to use in calculating radiative transfer (i.e. the "`nlf`" parameter in GRASIL);

`$dataSet->{'grasilOptions'}->{'radialGridCount'}` Specifies the number of radial grid cells to use in calculating radiative transfer (i.e. the "`ndr`" parameter in GRASIL).

[2] You can put these files where ever you want. Usually, we place them into `aux/Grasil/`.

`$dataSet->{'grasilOptions'}->{'recomputeSEDs}` If set to 1, SEDs will be computed for all galaxies even if they have been previously computed. (This can be useful to recompute SEDs with different options passed to GRASIL for example.) Set to 0 to re-use previously computed SEDs.

`$dataSet->{'grasilOptions'}->{'maxThreads}` Specifies the number of parallel threads to launch, each of which will run an instance of GRASIL. If not specified this number will default to the number of available cores.

`$dataSet->{'grasilOptions'}->{'cpuLimit}` Specifies the maximum time (in seconds) for which a Grasil calculation should be allowed to run before being terminated. Defaults to 3600s.

If necessary, the GRASIL code and data files will be downloaded automatically. Where possible, multiple instances of GRASIL are run in parallel to speed up the calculation.

The computed spectral energy distribution (SED) is stored in the HDF5 output file in dataset `grasilSEDs/Output<outputIn` where `outputIndeex`, `treeIndex` and `nodeIndex` are respectively the indices of the output, merger tree and node to which the galaxy belongs. The wavelengths and inclinations at which the SED is tabulated are similarly stored in the same group in datasets `wavelength` and `inclination`. If the SED has been previously computed for a given galaxy, it will be read from file instead of recomputing using `Grasil`. The flux is found by interpolating to the relevant rest-frame wavelength and osberved inclination.

The `Galacticus::Grasil` module supports the `selection` element of `$model`. If this element is set to contain a PDL giving the selection of galaxies to process then only those galaxies will have their `Grasil` fluxes computed, rather than all galaxies in the output. Note that if the resulting dataset is stored back to the HDF5 file then any non-selected galaxies will be assigned zero flux, and these zero fluxes will be reported on future attempts to access the flux[3].

2.6 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 GALACTICUSThis 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.

[3]The `selection` element of `$model` will be more generally supported in future versions of GALACTICUS and will more elegantly handle storing of partial datasets to file to avoid this problem

2.7 Perl Statistics Modules

Galacticus provides some Perl modules which compute useful statistics. These are described below.

2.7.1 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:

```
(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 σ 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.

2.8 On-The-Fly Analysis

Galacticus can perform various analyses on-the-fly (i.e. as it is running), outputting the expectation for a particular observed dataset. Analyses are selected by adding the appropriate label to the `[mergerTreeAnalyses]` parameter (multiple, space-separated labels can be added to this parameter). Available on-the-fly analyses are described below.

On-the-fly analyses exist for several different stellar and gas mass functions. These analyses share several common features which are described below. In general, the model masses are used to construct a mass function by binning into a histogram using the same bins as the observational dataset as the centers of the bins (with bin boundaries placed at the geometric means of consecutive bin centers), and with each galaxy contributing a weight equal to the volume weight of its merger tree. Typically the bins will be uniformly spaced in the logarithm of mass. Before accumulation to the histogram, galaxy masses are adjusted to account for two effects:

1. *Cosmological parameters:* Typically the observational data will have been analyzed assuming some specific set of cosmological parameters which will differ from that in the current model. Therefore, the mass of the galaxy and the weight assigned to it are both adjusted to match what would be inferred if they were assessed using the same cosmological parameters as were used for the observational data. Typically, this will mean that masses are scaled in proportion to $D_\mathrm{L}^2(\bar{z})$, where $D_\mathrm{L}(z)$ is the luminosity distance to redshift z and $\bar{z}$ is the median redshift of observational sample, while weights are scaled in proportional to $D_\mathrm{c}^2 H^{-1}(z)$, where $D_\mathrm{c}(z)$ is the comoving distance to redshift z and $H(z)$ is the Hubble parameter at redshift z. These scalings are typical for galaxies where masses are determined from luminosities, but may vary in other cases.

2. *Systematic errors:* Each mass function allows for a systematic shift in model masses (to account for systematic uncertainties in the observational analysis) using a simple model. Specifically, model masses are mapped by this model as follows

$$\log_{10} M \rightarrow \log_{10} M + \sum_{i=0}^{N} \alpha_i \log_{10}^i (M/M_0), \tag{2.9}$$

   where M_0 is a mass scale defined for each analysis, N is fixed integer for each analysis, and the coefficients α_i are input parameters `[<label>MassSystematic<i>]`, where `<label>` is the label of the analysis and `<i>` is an integer in the range $0 \ldots N$.

Individual analyses may defined additional transformations of the galaxy mass. These are detailed below.

When the weight of each galaxy is accumulated to the mass function histogram, the logarithm of the galaxy mass is modeled as a Gaussian kernel with width specified by each analysis to account for random errors in the observations and/or scatter in model masses. That is, the weight of each galaxy is spread over every bin of the histogram using this Gaussian kernel. Additionally, if `[analysisMassFunctionsApplyGravitationalLensing` then the mass of each galaxy is convolved with the magnification distribution expected due to gravitational lensing from large-scale structure (see §13.19).

The contribution to the mass function from each model output redshift is computed from the volume associated with that output redshift, given the angular geometry and depth of the survey as determined from the appropriate survey geometry (see §13.59) and assuming that each output redshift represents a range of redshifts running between the geometric means of the times corresponding to each output redshift.

In addition to the mass function, the covariance matrix, $\mathbf{C}_\mathrm{model}$, of the mass function is also computed. The assumptions used when constructing the covariance matrix are controlled by the parameter `[analysisMassFunctionCovarianceModel]`. If set to `binomial`, them to construct $\mathbf{C}_\mathrm{model}$ we make use of the fact that Galacticus works by sampling a set of tree "root masses" from the $z = 0$ dark matter halo mass function. From each root, a tree is grown, within which the physics of galaxy formation is then solved. Root masses are sampled uniformly from the halo mass function. That is, the cumulative halo mass function, $N(M)$, is constructed between the maximum and minimum halo masses to be simulated. The number of root masses, N_r, to be used in a model evaluation is then determined. Root masses are then chosen such that

$$N(M_i) = N(M_\mathrm{min}) \frac{i-1}{N_\mathrm{r}-1} \tag{2.10}$$

for $i = 1 \ldots N_\mathrm{r}$ (noting that $N(M_\mathrm{max}) = 0$ by construction).

Consider first those galaxies which form in the main branch of each tree (i.e. those galaxies which are destined to become the central galaxy of the $z = 0$ halo). Suppose that we simulate N_k halos of root mass M_k at $z = 0$. In such halos the main branch galaxies will, at any time, have stellar masses drawn from some distribution $p_k(M_\star|t)$. The number of such galaxies contributing to bin i of the mass function

is therefore binomially distributed with success probability $p_{ik} = \int_{M_{i,min}}^{M_{i,max}} p_k(M_\star|t)\mathrm{d}M_\star$ and a sample size of N_k. The contribution to the covariance matrix from these main branch galaxies is therefore:

$$\mathcal{C}_{ij} = \begin{cases} p_{ik}(1-p_{ik})N_k w_k^2 & \text{if } i = j \\ -p_{ik}p_{jk}N_k w_k^2 & \text{otherwise,} \end{cases} \tag{2.11}$$

where w_k is the weight to be assigned to each tree. To compute this covariance requires knowledge of the probabilities, p_{ik}. We estimate these directly from the model. To do this, we bin trees into narrow bins of root mass and assume that p_{ik} does not vary significantly across the mass range of each bin. Using all realizations of trees that fall within a given bin, k, we can directly estimate p_{ik}. In computing p_{ik}, the range of halo masses considered and the fineness of binning in halo mass are determined by the parameters `[analysisMassFunctionsHaloMassMinimum]`, `[analysisMassFunctionsHaloMassMaximum]`, and `[analysisMassFunctionsHaloMassBinsPerDecade]`.

If instead, `[analysisMassFunctionCovarianceModel]=Poisson`, the main branch galaxies are modeled as being sampled from a Poisson distribution (and so off-diagonal terms in the covariance matrix will be zero).

In addition to the main branch galaxies, each tree will contain a number of other galaxies (these will be "satellite" galaxies at $z = 0$, but at higher redshifts may still be central galaxies in their own halos). Tests have established that the number of satellites in halos is well described by a Poisson process. Note that, as described above, each galaxy contributes a Gaussian distribution to the mass function due to modelling of random errors in stellar mass determinations. For main branch galaxies this is simply accounted for when accumulating the probabilities, p_{ik}. For satellite galaxies, off-diagonal contributions to the covariance matrix arise as a result, $C_{ij} = w_k f_i f_j$, where f_i is the fraction of the galaxy contributing to bin i of the mass function.

The parameter `[analysisMassFunctionsCorrelationTruncateLevel]` allows off-diagonal elements in the model covariance matrix whose correlation is less than the specified value to be truncated to zero. This helps avoid remove numerical noise from the covariance matrix.

2.8.1 ALFALFA HI Mass Function

This analysis, selected using label `alfalfaHiMassFunctionZ0.00`, computes the model expectation for the HI gas mass function of Martin et al. [2010]. Calculation of the mass function follows the basic methdology outlined above. Model galaxy masses and weights are mapped for differences in cosmological parameters as described above, and the simple systematic mass error model is employed with parameter $N = 1$ and $M_0 = 10^9 M_\odot$.

The analysis assumes that only the total interstellar medium (ISM) mass of each galaxy is available, along with the disk radius (assuming an exponential disk). To infer the HI mass the model of Obreschkow et al. [2009] is used. Specifically, the molecular ratio, $R_{\rm mol} \equiv M_{\rm H_2}/M_{\rm HI}$, is given by:

$$R_{\rm mol} = \left(A_1 R_{\rm mol}^{{\rm c}\,\alpha_1} + A_2 R_{\rm mol}^{{\rm c}\,\alpha_2}\right)^{-1}, \tag{2.12}$$

where the ratio at the disk center is given by

$$R_{\rm mol}^{\rm c} = [K r_{\rm disk}^{-4} M_{\rm gas}(M_{\rm gas} + \langle f_\sigma \rangle M_\star)]^\beta. \tag{2.13}$$

Here, $R_{\rm mol}$ is the mass ratio of H_2 to HI, $M_\star$ is the stellar mass of the disk, $r_{\rm disk}$ is the disk exponential scale length, $\langle f_\sigma \rangle$ is the average ratio of the vertical velocity dispersions of gas to stars, and $K = {\rm G}/(8\pi P_\star)$. The HI mass is then determined from:

$$M_{\rm HI} = X_{\rm H} M_{\rm gas}/(1 + R_{\rm mol}), \tag{2.14}$$

where $X_{\rm H} = 0.778$ is the primordial hydrogen fraction by mass. In the above $K =$`[alfalfaHiMassFunctionZ0.00MolecularF`
$\langle f_\sigma \rangle =$`[alfalfaHiMassFunctionZ0.00MolecularFractionfSigma]`, $A_1 =$`[alfalfaHiMassFunctionZ0.00MolecularFracti`

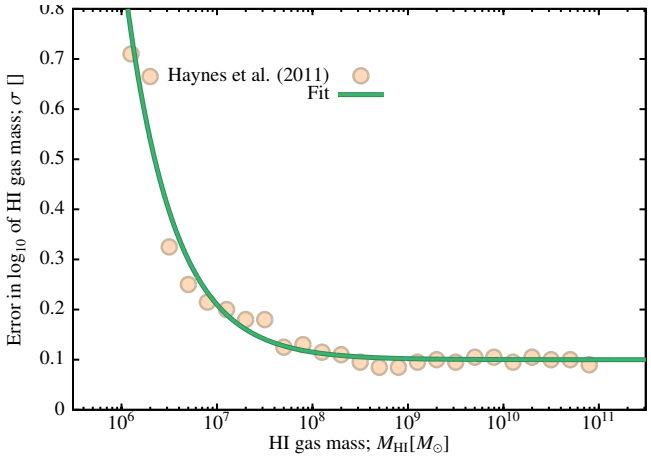


Figure 2.2: The observational random error in galaxy HI mass as a function of HI mass for the ALFALFA survey. Points show the errors reported by Haynes et al. [2011], while the line shows a simple functional form fit to these errors.

$A_2 =$`[alfalfaHiMassFunctionZ0.00MolecularFractionA2]`, $\alpha_1 =$`[alfalfaHiMassFunctionZ0.00MolecularFractionAlp`
$\alpha_2 =$`[alfalfaHiMassFunctionZ0.00MolecularFractionAlpha2]`, and $\beta =$`[alfalfaHiMassFunctionZ0.00MolecularFract`
Default values for these parameters are taken from Obreschkow et al. [2009]. According to Obreschkow (private communication), there remains significant scatter of $\sigma_{R_{\rm mol}} = 0.4$ dex between the predicted $R_{\rm mol}$ from this model and that observed. This is accounted for in when constructing the mass function (see below).

To account for both observational errors and scatter in $R_{\rm mol}$ not captured by the above model, the HI mass of each galaxy is modeled as a Gaussian in $\log_{10} M_{\rm HI}$ when constructing the mass function. Observational random errors on HI mass, including those arising from flux density uncertainties and errors in the assumed distance to each source, are taken from Fig. 19 of Haynes et al. [2011]. The magnitude of the error as a function of HI mass is fit using a functional form:

$$\sigma_{\rm obs} = a + \exp\left(-\frac{\log_{10}(M_{\rm HI}/M_\odot) - b}{c}\right), \tag{2.15}$$

where $\sigma_{\rm obs}$ is the error on $\log_{10}(M_{\rm HI}/M_\odot)$. We find a reasonable fit using values[4] of $a = 0.100 \pm 0.010$, $b = 5.885 \pm 0.100$, and $c = 0.505 \pm 0.020$ as shown in Fig. 2.8.1. The total random error on the logarithm of each galaxy mass is given by $\sigma^2 = \sigma^2_{R_{\rm mol}} + \sigma^2_{\rm obs}$, and is used as the width of the Gaussian kernal when applying each galaxy to the mass function histogram (as described above).

Additionally, HI mass estimates can be affected by HI self-absorption for highly inclined galaxies. [Zwaan et al., 1997, see also Zwaan et al. 2005] estimate that this effect would lead to a mean underestimation of HI masses by a factor 1.1 for a randomly oriented galaxy sample. Therefore, a value of -0.0414 for the systematic parameter `[alfalfaHiMassFunctionZ0.00MassSystematic0]` is recommended.

[4]This should not be regarded as a formal good fit. Error estimates are approximate—we have simply found a functional form that roughly describes them, along with conservative errors on the parameters of this function which are included in the priors.

3 Plotting Support

3.1 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:

```
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...

```
&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 `epslatex` terminal along with `color` and `colortext` options, any line weight preferences and the output EPS file. This is followed by commands to set up the plot, including labels, ranges etc. Note that you *must* specify margins manually[2]. Following this are calls to `&PrettyPlots::Prepare_Dataset` which prepares instructions for plotting of a single dataset. The first argument is a reference to a structure which will store the instructions, while the second and third arguments are PDLs containing the x and y data to be plotted. Following this are multiple options as follows:

`title` Gives the title of the dataset for inclusion in the plot key;

`style` Specifies how the dataset should be drawn: either `line`, `point`, `boxes`, or `filledCurve`;

`linePattern` Specifies the line pattern (as defined for GnuPlot's `lt` option) to use;

`symbol` A two element list giving the symbol indices that should be used to plot the border and inner parts of each point respectively;

`weight` A two element list giving the line weights to be used for border and inner parts of each point/line respectively;

`color` A two element list giving the color of border and inner parts of each point/line respectively. Colors should be specified as `#RRGGBB` in hexadecimal. Several suitable color pairs and sequences of pairs are defined in the `GnuPlot::PrettyPlots` module;

`pointSize` Specifies the size of the points to be used;

`errorNNN` Gives a PDL containing sizes of errors to be plotted on points in the up, down, left and right directions. A zero value will cause the error bar to be omitted, while a negative value will cause an arrow to be drawn with a length equal to the absolute value of the specified value;

`filledCurve` If the `filledCurve` style is used, this option specifies the type of filled curve (`closed`, `x1`, `x2`, etc.—see the GnuPlot `help filledcurve` text for complete options). The default is `closed`;

`y2` If the `filledCurve` style is used along with the `filledCurve=closed` option, this option is used to specify a second PDL of y-axis values. The region between this curve and the usual y-axis curve will be filled.

[2] The `GnuPlot::PrettyPlots` module works by generating multiple layers of plotting which are overlaid. Axes are only drawn for the first layer. If you do not specify margins manually, they will be computed automatically for each layer and so will not match up between all layers. This will result in data being plotted incorrectly.

Once all datasets have been prepared, the call to `&PrettyPlots::Plot_Datasets` will generate the EPS and LaTeX files necessary to make the plot. This resulting plot can be converted to PDF, PNG or ODG form by calling `&LaTeX::GnuPlot2PDF`, `&LaTeX::GnuPlot2PNG` or `&LaTeX::GnuPlot2ODG` respectively. The EPS file will be replaced with the appropriate file. The `&LaTeX::GnuPlot2PNG` routine accepts an optional `backgroundColor` argument in `#RRGGBB` format. If present, this color will be used to set the background color of the plot (otherwise white is assumed). Although the background is made transparent in the PNG, setting the background color is important as antialiasing will make use of this background. Note that both PNG and ODG options will switch black axes and labels to white[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`).

3.2 Merger Tree Diagrams with DOT

The DOT command, which is a part of GraphViz is useful for creating diagrams of merger trees. Galacticus provides a function to output the structure of any merger tree in GraphViz format. This function, `Merger_Tree_Dump`, is provided by the `Merger_Trees_Dump` module. Usage is as follows:

```
call Merger_Tree_Dump(                                    &
     &                index                               , &
     &                baseNode                            , &
     &                highlightNodes      =highlightNodes , &
     &                backgroundColor     ='white'        , &
     &                nodeColor           ='black'        , &
     &                highlightColor      ='black'        , &
     &                edgeColor           ='#DDDDDD'      , &
     &                nodeStyle           ='solid'        , &
     &                highlightStyle      ='filled'       , &
     &                edgeStyle           ='solid'        , &
     &                labelNodes          =.false.        , &
     &                scaleNodesByLogMass=.true.          , &
     &                edgeLengthsToTimes  =.true.         , &
     &                path                ='/my/path'       &
     &               )
```

Here `index` is the tree index (successive calls to `Merger_Tree_Dump` with the same index will result in a sequence of output files—see below), and `baseNode` is a pointer to the base node of the tree to be dumped. All other arguments are optional:

`highlightNodes` A list of node IDs. All nodes listed will be highlighted in the diagram;

`backgroundColor` The color for the background of the diagram;

`nodeColor` The color used to draw nodes;

`highlightColor` The color used for highlighted nodes;

`edgeColor` The color of edges (lines joining nodes);

`nodeStyle` The style to use when drawing nodes;

[3] This is just a presonal preference for plots displayed in presentations—other options could be added

`highlightStyle` The style to use when drawing highlighted nodes;

`edgeStyle` The style to use when drawing edges;

`labelNodes` Specifies whether or not nodes should be labelled (labels consist of the node ID followed by the redshift);

`scaleNodesByLogMass` If true, the size of nodes will be set to be proportional to the logarithm of the node mass;

`edgeLengthsToTimes` If true, the spacing between parent and child nodes will be proportional to the logarithmic time interval between them.

`path` If present, write tree dumps into this directory. Otherwise, the current directory will be used.

All colors and styles are character strings and can be in any format understood by DOT. The tree structure will be dumped to file named `mergerTreeDump:`⟨`ID`⟩`:`⟨`N`⟩`.gv` where ⟨`ID`⟩ is the index of the tree and ⟨`N`⟩ increasing incrementally from 1 each time the same tree is consecutively dumped. These files can be processed using DOT. For example

```
dot -Tps mergerTreeDump:1:1.gv > tree.ps
```

will create a tree diagram as the PostScript file `tree.ps`.

4 Tutorials

This chapter contains step-by-step guides to performing common tasks with Galacticus.

4.1 Running Galacticus on N-body Merger Trees

See §A.11 for details of how to build merger tree files suitable for input into Galacticus. There are many options which control precisely how merger trees read from file should be handled. The following section provides guidance on the best choice of parameters.

4.1.1 Setting Input Parameters

To utilize merger trees from the file[1] that you created in a Galacticus run it's necessary to set two parameters in the input parameter file that you will use for the run:

```
<!-- Specify that merger trees are to be read from file, and give the name of the file to read -->
<mergerTreeConstructMethod value="read"             />
<mergerTreeReadFileName    value="myNBodyTrees.hdf5"/>
```

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 order to choose sensible settings for the various parameters that control merger trees read from file, it is recommended that you read through each of the items below and follow the guidance given.

Cosmology: In addition to specifying that trees should be read from a file, it's also important to ensure that the values of cosmological parameters in Galacticus match those in the merger tree file. (If they don't match, Galacticus will stop with an error message unless you set `[mergerTreeReadMismatchIsFatal]=false` in which case you'll just be warned about any mismatch.) In our case of using merger trees from the Millennium Simulation, the correct cosmological parameter values can be set as follows:

```
<!-- Use Millennium Simulation cosmology. -->
<cosmologyParametersMethod value="simple"/>
 <HubbleConstant  value="73.0"  />
 <OmegaMatter     value="0.25"  />
 <OmegaDarkEnergy value="0.75"  />
 <OmegaBaryon     value="0.0455"/>
</cosmologyParametersMethod>
<cosmologicalMassVarianceMethod value="filteredPower">
  <sigma_8 value="0.900"/>
</cosmologicalMassVarianceMethod>
<powerSpectrumPrimordialMethod value="powerLaw">
  <index               value="1.000"/>
  <wavenumberReference value="1.000"/>
```

[1] The following assumes that merger trees will be read from a file following Galacticus's standard HDF5 format which is described in Appendix A.

```
    <running              value="0.000"/>
  </powerSpectrumPrimordialMethod>
```

Existance at Final Time: 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:

```
<allTreesExistAtFinalTime value="false"/>
```

Snapping Nodes to Snapshots: 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:

```
<mergerTreeReadOutputTimeSnapTolerance value="1.0d-3"/>
```

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).

Missing Hosts: Galacticus expects to find each node's host node present in a merger tree forest. If a node's host is not found this is cause for a fatal error to be issued, since it is impossible to correctly construct and evolve the corresponding forest. If you absolutely want to run a forest for which one or more host nodes are missing, you can allow this by setting `[mergerTreeReadMissingHostsAreFatal]=false`—in this case missing host nodes trigger a warning only and nodes without a host are forced to become isolated nodes. This will lead to incorrect tree evolution however, so the recommended setting is:

```
<mergerTreeReadMissingHostsAreFatal value="true"/>
```

Branch Jumps and Subhalo Promotions: If your merger trees contain subhalos they will most likely exhibit two specific behaviors[2]: i) nodes which are subhalos in one timestep may become non-subhalos (isolated halos) in a subsequent timestep ("subhalo promotion"), and ii) nodes which are subhalos in one branch of the tree may "jump" to another branch[3] of the tree becoming a subhalo there ("branch jumping"). These behaviors are fully supported by Galacticus and so we recommend the following settings:

```
<mergerTreeReadAllowSubhaloPromotions value="true"/>
<mergerTreeReadAllowBranchJumps       value="true"/>
```

You may choose to disallow these behaviors by setting either of the above parameters to `false`—for example if you wish to explore how your results would differ if subhalos were forced to remain subhalos forever in their original branch. Note that allowing subhalo promotion while not allowing branch jumping can lead to deadlocks in merger tree evolution, so change these settings with caution.

Note that for trees which do not contain subhalos these two parameters are irrelevant.

Subhalo Masses: If your trees contain subhalos, the mass evolution of those subhalos can be preset in the satellite component of each node. In this way, the subhalo mass in Galacticus will track that specified by the merger tree file. This requires the use of a satellite component which allows presetting of subhalo masses. Recommended settings are therefore:

```
<treeNodeMethodSatellite           value="preset"/>
<mergerTreeReadPresetSubhaloMasses value="true"  />
```

If your trees do not contain subhalos, recommended settings are instead:

[2] These two behaviors are called out as they specifically *do not* occur in merger trees created using Press-Schechter-based algorithms for example.

[3] That is, the subhalo's descendented is hosted by a node other than the descendent of the subhalo's host.

```
<treeNodeMethodSatellite           value="standard"/>
<mergerTreeReadPresetSubhaloMasses value="false"   />
```

Halo Positions/Velocities: If your trees contain position and velocity information for halos, those positions and velocities can be preset in the position component of each node. This requires the use of a position component which allows presetting of positions and velocities. Recommended settings are therefore:

```
<treeNodeMethodPosition        value="preset"/>
<mergerTreeReadPresetPositions value="true"  />
```

If your trees do not contain position information recommended settings are:

```
<treeNodeMethodPosition        value="null" />
<mergerTreeReadPresetPositions value="false"/>
```

Subhalo Orbits: If your trees contain position and velocity information they can be used to preset initial orbit information for subhalos. Note that it is not required that your trees contain subhalos for this orbit presetting to be performed—Galacticus can follow subhalo orbits even if subhalos are not included in the trees themselves. The following settings are recommended:

```
<mergerTreeReadPresetOrbits             value="true"/>
<mergerTreeReadPresetOrbitsSetAll       value="true"/>
<mergerTreeReadPresetOrbitsAssertAllSet value="true"/>
<mergerTreeReadPresetOrbitsBoundOnly    value="true"/>
```

These options will cause an orbit to be preset for each subhalo based on the relative position and velocity of merging halos and assuming that the orbital energy and angular momentum are conserved between the time immediately prior to the merger and the time of virial radius crossing. If the computed orbit does not cross the virial radius of the larger halo or if the computed orbit is unbound, the above options cause an orbit to be preset by drawing orbital parameters at random from the chosen cosmological distribution (see §13.51.2).

Subhalo Merging: If your merger trees contain subhalo information, that information can be used to specify when, and with which other node, each subhalo merges. Specifically, a subhalo is assumed to merge at the time at which it is not the primary progenitor of its descendent halo—possibly with some other delay to be described below. Recommended settings are:

```
<mergerTreeReadPresetMergerTimes          value="true"             />
<mergerTreeReadPresetMergerNodes          value="true"             />
<mergerTreeReadSubresolutionMergingMethod value="boylanKolchin2008"/>
```

The first two options cause subhalos to merge at the time described above, and with their descendent node. The final option accounts for the possibility that the subhalo should not actually merge immediately at this time. For example, in N-body simulations, the subhalo may have simply been lost due to limitations of resolution or halo finder algorithms. The final option specifies that some additional time until merging be added based on the subhalo merging timescale algorithm of Boylan-Kolchin et al. [2008; see §13.51.1], and computed using the last known orbital properties of the subhalo.

Halo Scale Radii: If your merger trees contain information on halo scale radii or half-mass radii, these can be used to preset the scale radius of each node. This requires the use of a dark matter profile component which allows presetting of scale length. Recommended settings are therefore:

```
<mergerTreeReadPresetScaleRadii                  value="true"     />
<mergerTreeReadPresetScaleRadiiFailureIsFatal    value="true"     />
```

```
<mergerTreeReadPresetScaleRadiiConcentrationMinimum value="3"        />
<mergerTreeReadPresetScaleRadiiConcentrationMaximum value="60"       />
<mergerTreeReadPresetScaleRadiiMinimumMass          value="see below"/>
```

Minimum and maximum concentrations are specified—these are used to restrict the range of scale radii that are allowed for a given halo. If scale radii are to be determined based on half-mass radii given in the merger tree file, and if the computed scale radius does not result in a concentration between these limits, then a fatal error is issued.

Finally, you can set a minimum halo mass via the `[mergerTreeReadPresetScaleRadiiMinimumMass]` parameter below which the scale radii or half-mass radii in your file should be considered not reliable. For halos below this mass, scale radii will instead be assigned via the selected dark matter halo concentration method (see §13.11.3).

Halo Angular Momenta: If your merger trees contain spin or angular momentum information these can be preset for each node. Recommended settings are:

```
<treeNodeMethodSpin                  value="preset"/>
<mergerTreeReadPresetSpins           value="true"  />
<mergerTreeReadPresetUnphysicalSpins value="true"  />
```

The last of these options causes any halos for which the spin given in the merger tree file is non-positive to be assigned a spin at random instead, drawn from the specified cosmological distribution (see §13.11.7).

If subhalo masses are not included in their host halo masses in your merger tree file, you should specify how the angular momenta of subhalos should be accounted for when adding their mass to their host halo. If positions and 3D angular momenta are available in your merger tree file, the recommended setting is:

```
<mergerTreeReadSubhaloAngularMomentaMethod value="summation"/>
```

If this information is not present

```
<mergerTreeReadSubhaloAngularMomentaMethod value="scale"    />
```

should be used instead.

If your merger tree file contains 3D spin or angular momentum information, you can choose to make that information available within Galacticus be using the settings:

```
<treeNodeMethodSpin          value="preset3D"/>
<mergerTreeReadPresetSpins3D value="true"    />
```

Subhalo Indices: If your merger trees contain subhalos, you can tell Galacticus to keep track of the indices of subhalos by setting:

```
<treeNodeMethodSatellite            value="preset"/>
<mergerTreeReadPresetSubhaloIndices value="true"  />
```

The Galacticus output file will then contain `satelliteNodeIndex` datasets which list the index (as given in the merger tree file) for all subhalos and halos. Without specifying this presetting, the index of subhalos is frozen at the index of the halo immediately prior to it becomming a subhalo.

The remainder of this section gives more detail about many of the parameters described above and how they affect handling of merger trees read from file. Further parameters can be set to control what information from the stored trees will be used in Galacticus. Examples are given below.

4.1.2 Further Details

Further details of the effects of the many parameters controlling merger trees read from file 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:

```
<!-- Use merger tree node positions -->
<treeNodeMethodPosition        value="preset"/>
<mergerTreeReadPresetPositions value="true"  />
```

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 -->
<treeNodeMethodPosition        value="null" />
<mergerTreeReadPresetPositions value="false"/>
```

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 -->
<mergerTreeReadPresetOrbits             value="true"/>
<mergerTreeReadPresetOrbitsBoundOnly    value="true"/>
<mergerTreeReadPresetOrbitsSetAll       value="true"/>
<mergerTreeReadPresetOrbitsAssertAllSet value="true"/>
```

Typically, a merging halo is not seen at precisely the time at which it crosses the virial radius of its host (due to the fact that N-body simulations are output at discretely spaced timesteps). Therefore, Galacticus computes the orbit at the time just prior to merging and assumes that the orbital parameters (energy and angular momentum) remain fixed to propagate the orbit to the virial radius of the host. The second parameter in the above example, `[mergerTreeReadPresetOrbitsBoundOnly]`, specifies whether or not only bound orbits should be set. Some calculations (e.g. of subhalo merging times) assume bound orbits and may fail if given an unbound orbit. Setting this option to `true` causes only bound orbits to be preset—unbound orbits are ignored. Note that some orbits cannot be propagated to the virial radius (i.e. their pericenter is larger than the virial radius). The `[mergerTreeReadPresetOrbitsSetAll]` option, if true, will cause such orbits to be assigned randomly using the selected `[virialOrbitsMethod]`, such that all orbits are assigned. The `[mergerTreeReadPresetOrbitsAssertAllSet]` option requires that all orbits be set—if `[mergerTreeReadPresetOrbitsSetAll]`=`false` and `[mergerTreeReadPresetOrbitsAssertAllSet]`=`true` then Galacticus will exit with an error message if any orbit cannot be set.

If the satellite component additionally permits setting of the satellite position and velocity, these properties will also be assigned based on the relative position and velocity of the satellite and host halos.

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

```
<treeNodeMethodSatellite         value="preset"/>
<mergerTreeReadPresetMergerTimes value="true"  />
```

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,

```
<treeNodeMethodSatellite         value="standard" />
<mergerTreeReadPresetMergerNodes value="false"    />
<satelliteMergingMethod          value="Jiang2008"/>
```

which selects a standard satellite orbit method, prevents attempts to preset the merger times and selects the `Jiang2008` method for computing merger times instead.

In addition to setting the times of merger events, it is possible to set the target node with which a merging node should merge. By default, Galacticus will assume that all merging occurs with the non-subhalo host node in which a subhalo is located. This may not be the desired behavior when using N-body merger trees. For example, such trees may indicate that a subhalo merges with another subhalo. Setting

```
<mergerTreeReadPresetMergerNodes value="true"/>
```

will cause the target node with which each merger should occur to be determined from the merger tree structure and preset for use in Galacticus.

It is possible to add a delay between the last time at which a subhalo was seen in a simulation and the time at which it is considered to merge. This functionality is motivated by the consideration that a subhalo vanishing from a simulation may be simply due to it dropping below resolution rather than it actually having undergone a merger. The parameter `[mergerTreeReadSubresolutionMergingMethod]` can be used to select a satellite merging timescale method (see §13.51.1) to use in this case. (It is set by default to "`null`" such that no delay before merging occurs.) The orbit of the subhalo around its parent at the last time it is present in the merger tree is passed to this method and used to estimate a time until merging. This delay is added to the time at which the subhalo merges and, if merge target nodes are being set, the target node is updated accordingly.

Subhalo Indices

The indices of subhalos are usually frozen at the index of the halo just prior to becoming a subhalo. The index of the corresponding halo in the original tree (as read from file) can be tracked as follows:

```
<treeNodeMethodSatellite            value="preset"/>
<mergerTreeReadPresetSubhaloIndices value="true"  />
```

to first select the "preset" satellite orbit method (which allows subhalo indices to be preset) and, second, to instruct the merger tree construction algorithm to preset those indices. The index will then be available in output as `satelliteNodeIndex`.

Subhalo Masses

The masses of subhalos (specifically their time evolution after they become subhalos) can be set using the values stored in the merger tree file (if available). To set subhalo masses in this way use

```
<treeNodeMethodSatellite           value="preset"/>
<mergerTreeReadPresetSubhaloMasses value="true"  />
```

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[4] by setting:

```
<mergerTreeReadPresetSpins value="true"/>
```

The spin parameter is set using the spin of each node if available, or otherwise using the angular momentum of each node stored in the merger tree file using:

$$\lambda = \frac{|\mathbf{J}||E|^{1/2}}{\mathrm{G}M^{5/2}} \tag{4.1}$$

where $|\mathbf{J}|$ is the magnitude of the node's angular momentum, M is the node's mass and E is its energy. Additionally, by setting:

```
<mergerTreeReadPresetSpins3D value="true"/>
```

the spin vector of each node will be set (assuming that the vector spin or angular momenta of nodes are available in the merger tree file) using:

$$\lambda = \frac{\mathbf{J}|E|^{1/2}}{\mathrm{G}M^{5/2}}. \tag{4.2}$$

If spins could not be determined for some halos the spin (or angular momentum) should be set to zero in the merger tree file, and the parameter `[mergerTreeReadPresetUnphysicalSpins]` set to `true`. Galacticus will then assign a spin to such halos by sampling from the selected spin distribution (see §13.11.7).

Node Scale Radii

If information on the half-mass or scale radii of nodes is available in the merger tree file, it can be used to preset the value of the dark matter halo scale radius in each node by setting:

```
<mergerTreeReadPresetScaleRadii value="true"/>
```

Before doing this, it is important to be sure that the half-mass or scale radii of the nodes are reliable. For example, in low mass nodes extracted from an N-body simulation resolution effect may limit the accuracy of the measured half-mass or scale radius. In such cases, use the `[mergerTreeReadPresetScaleRadiiMinimumMass]` parameter to specify the lowest mass halos for which the scale radii should be preset—lower mass halos will be assigned a scale radius using the method specified by the `[mergerTreeReadConcentrationFallbackMethod]` parameter (which will default to the value of `[darkMatterConcentrationMethod]`; see §13.11.3). It is also possible to specify minimum and maximum allowed concentrations when computing the scale radius from the half mass radius using the `[mergerTreeReadPresetScaleRadiiConcentrationMinimum]` and `[mergerTreeReadPresetScaleRadiiConcentrationMaximum]` parameters. If matching the half mass radius would require a concentration outside of this range, Galacticus will abort unless `[mergerTreeReadPresetScaleRadiiFa` in which case it will instead silently use the fallback concentration method described above.

If only half-mass radii are available, the scale radius is set by using a root finding algorithm to ensure that half of the total halo mass is enclosed within the specified half-mass radius.

[4] 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.

Miscellaneous N-body Properties

Several miscellaneous properties often available from N-body merger trees can also be preset by setting the following parameters to `true`:

mergerTreeReadPresetParticleCounts Sets the number of particles in each halo (requires the `particleCount` dataset to be present in the merger tree file);

mergerTreeReadPresetVelocityMaxima Sets the maxima of halo rotation curves (requires the `velocityMaximum` dataset to be present in the merger tree file);

mergerTreeReadPresetVelocityDispersions Sets the velocity dispersion of halos (requires the `velocityDispersion` dataset to be present in the merger tree file).

Subhalo Promotion

A subhalo may, at a later time, become an isolated halo once again. Galacticus allows you to control whether such behavior is allowed, or should be prohibited. To allow such "subhalo promotion", set:

```
<mergerTreeReadAllowSubhaloPromotions value="true"/>
```

If you choose to inhibit this behavior by setting the above parameter to false, a halo that becomes a subhalo will remain a subhalo forever thereafter. Note that the isolated halo to which it would have been promoted will still exist, and may therefore form its own galaxy. This can result in double counting of mass, and so inhibiting subhalo promotion is not recommended.

"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). 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 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:

```
<mergerTreeReadMissingHostsAreFatal value="false"/>
```

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.

4.1.3 Using Particles to Track Unresolved Subhalos

In N-body simulations it is possible that a subhalo can become "lost" from the simulation (i.e. can no longer be identified by a halo finder due to resolution issues) before it has actually merged with the central galaxy or been completely tidally destroyed. In such cases it is useful to be able to assign a position to the subhalo at later times. A common approach to assigning a position (and velocity) is to use that of the most bound particle in the subhalo at the last time it was identified. Galacticus allows for particle tracking in this way through the addition of particle information to the merger tree file.

To add particle tracking data to a merger tree file, follow these steps:

1. Identify all subhalos which are lost from the simulation prior to the final timestep;

2. Determine the index of the most bound particle in each such subhalo in the last timestep in which it was identified;

3. For each subhalo, extract the redshift, position, and velocity of that particle (which is usally trivial to do once its index is known) at each subsequent timestep in the simulation;

4. Write these data (along with the particle index) to the `particles` group in the merger tree file as described in §A.10;

5. Add two datasets to the `forestHalos` group:

   a) `particleIndexStart` which should indicate the index in the datasets in the `particles` group at which the data for each halo begins (or -1 if no particle data is included for the halo);

   b) `particleIndexCount` which should indicate the number of entries in the datasets in the `particles` group for each halo (or -0 if no particle data is included for the halo).

4.1.4 Handling of Extremely Large Merger Tree Forests

Halos can move between merger trees (see §4.1.2 and §4.1.2), leading to the necessity of merger tree forests—interconnected groups of merger trees that Galacticus typically processes as a whole. These forests can become very large—in some cases so large that they do not fit within the available memory. Galacticus can handle such forests by splitting them into individual trees. Each tree is processed separately, and nodes are moved between trees as needed. If a tree needs a node from another tree before its evolution can continue, its state can be suspended to disk, and later re-read once the node it requires becomes available. In this way, very large forests can be processed without running out of memory (as trees are stored to disk while they are not being processed).

To cause forests to be split in this way, the following parameters should be set:

```
<treeEvolveSuspendToRAM                   value="false"            />
<treeEvolveSuspendPath                    value="/my/scratch/path/"/>
<mergerTreeReadForestSizeMaximum          value="10000000"         />
<mergerTreeReadSubresolutionMergingMethod value="infinite"         />
```

Here, `treeEvolveSuspendToRAM` specifies that merger trees should be suspended to disk (i.e. not to RAM which is the default), and `treeEvolveSuspendPath` gives a path where the suspended trees can be written—typically scratch space local to the compute node where Galacticus is being run is a good option.

`mergerTreeReadForestSizeMaximum` specifies the maximum number of nodes allowed in a forest before it will be split. A suitable number for this depends on the details of the available RAM, the number of threads sharing that RAM, and the characteristics of the Galacticus model being used (which will affect the memory required per node).

Finally, `mergerTreeReadSubresolutionMergingMethod` is set to `infinite` to prevent any merging (which is not supported for split forests at present, although it should be soon).

4.1.5 Analyzing the Output

Positions and Velocities

Components of the position of each node are output as `positionX`, `positionY` and `positionZ` and can be accessed in the same way as other output properties from Galacticus (see §2.1.7 and §2.2).

Subhalo Masses

The current mass of subhalos is available via the `nodeBoundMass` output dataset and can be accessed in the same way as other output properties from Galacticus (see §2.1.7 and §2.2). For non-subhalos this property is equal to the usual `nodeMass` property.

4.2 Generating Mock Catalogs with Lightcones from the Millennium Simulation

Suppose that you want to create a catalog of galaxies as would be found in a survey of an area of the sky out to some redshift. Such a "mock catalog" can be built by populating with galaxies all of the dark matter halos which happen to lie within the cone which that area makes as it is projected from the observer through the Universe.

Generating such a mock catalog using Galacticus involves first extracting the halos (and their merger trees) within this "lightcone" from a suitable N-body simulation, and then processing them through Galacticus. In this tutorial, we will specifically make use of the Millennium Simulation database to provide the merger trees, but the same principles apply to any N-body simulation.

The script, `scripts/aux/Millennium_Lightcone_Grab.pl` can be used to retrieve merger trees that intersect a given lightcone from the Millennium Database and to store them in Galacticus's format (see §A). The script is used as follows:

```
scripts/aux/Millennium_Lightcone_Grab.pl <lightconeDirectory> <fieldSize> <maximumRedshift>
    --user <myUserName> --password <myPassword> --treesPerFile <treesPerFile>
```

Here, `<lightconeDirectory>` is the name of a (pre-existing) directory into which merger tree data will be stored, `<fieldSize>` is the length (in degrees) of one side of the square field of view of the lightcone, `<maximumRedshift>` is the highist 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[5]

Each of the merger tree files created can then be run through Galacticus in the usual way (see §4.1). 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. -->
<outputRedshifts 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
    4.1795 4.5196 4.8884 5.2888 5.7239 6.1968"
```

[5] Note that these are not the ID numbers of the trees, just a sequential count of all trees retrieved.

```
/>

<!-- Add a lightcone filter with the required geometry -->
<mergerTreeOutput>
  <galacticFilterMethod value="lightcone"/>
</mergerTreeOutput>

<!-- Switch on output of lightcone data -->
<outputLightconeData value="true"/>

<!-- Prune away trees not appearing in the lightcone -->
<mergerTreeOperatorMethod value="pruneLightcone"/>

<!-- Specify lightcone geometry -->
<geometryLightconeMethod value="square">
  <origin value="0 0 0"/>
  <unitVector1 value=" 1 1  1"/>
  <unitVector2 value=" 0 1 -1"/>
  <unitVector3 value="-2 1  1"/>
  <lengthReplication value="500"/>
  <lengthHubbleExponent value="-1"/>
  <lengthUnitsInSI value="3.08567758135e22"/>
  <angularSize value="0.5"/>
  <timeEvolvesAlongLightcone value="true"/>
  <redshift 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
    4.1795 4.5196 4.8884 5.2888 5.7239 6.1968"
  />
</geometryLightconeMethod>
```

In the above `[outputLightconeData]` 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, along with the redshift) to be output (see §14.9), and `[mergerTreeOperatorMethod]` is set to `pruneLightcone` to cause any merger trees which have no nodes within the lightcone volume to be pruned away (as there is no need to process them). Finally, the `geometryLightconeMethod` parameter describes the geometry of the lightcone to be used—see §17.4.1 for details.

4.3 Using the Instantaneous Recycling Approximation

Choosing `[stellarPopulationPropertiesMethod]=instantaneous` will cause Galacticus to use the instantaneous recycling approximation for all calculations of stellar populations. The recyling rate and yield to use are set by the `[imfNAMERecycledInstantaneous]` and `[imfNAMEYieldInstantaneous]` parameters respectively, where `NAME` is the name of the appropriate initial mass function (IMF).

Setting `[stellarPopulationPropertiesMethod]=noninstantaneous` causes Galacticus to use a fully non-instantaneous, metal-depdendent calculation of recycling, metal production and supernovae

(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:

```
<!-- Force the calculation of recycling, yields etc. to   -->
<!-- be done assuming instantaneous recycling             -->
<starFormationImfInstantaneousApproximation value="true"/>
<!-- Set the mass of stars which should be used as the    -->
<!-- dividing line between long-lived and instantaneously -->
<!-- evolving in this approximation.                      -->
<starFormationImfInstantaneousApproximationMassLongLived value="1.0"/>
<!-- Set the effective age of populations to use in this -->
<!-- approximation when computing SNe numbers.           -->
<starFormationImfInstantaneousApproximationEffectiveAge value="13.8"/>
```

4.4 Computing Dust Attenuated Luminosities for All Galaxies

As described in §2.2.1, dust-attenuated luminosities can be computed for galaxies using the calculations of Ferrara et al. [1999] (among other methods). A script is provided which will automatically perform this calculation for all filters, in all galaxies, at all output redshifts in a Galacticus output file and store the results (along with the assumed galaxy inclinations) back to the HDF5 file. The script is used as follows:

```
./scripts/analysis/dustExtinguish.pl <galacticusFileName>
```

4.5 Computing Dust Attenuation and Emission Using Galacticus+Grasil

Galacticus can interface with the Grasil code to compute the attenutation of starlight by dust, along with the re-emission of absorbed energy by that dust. To do this, it is necessary to store the entire star formation history of galaxies in a Galacticus model, as Grasil uses this information to determine the attenuation of stellar populations as a function of their age.

Recording star formation histories is as simple as setting the `[starFormationHistoriesMethod]` parameter to `metallicitySplit`. This particular star formation history method stores the star formation in each galaxy as a function of time and metallicity, as required by Grasil. The level of detail with which the star formation history is stored is controlled by several parameters:

`starFormationHistoryTimeStep` The timestep used in discretizing star formation histories.

`starFormationHistoryFineTimeStep` The timestep to use in discretizing star formation histories just prior to output times. This should typically be smaller than `[starFormationHistoryTimeStep]` to give improved resolution in the star formation history for recently formed stars.

`starFormationHistoryFineTime` The period before each output for which the `[starFormationHistoryFineTimeStep]` should be used.

`starFormationHistoryMetallicityCount` The number of bins in metallicity to use when discretizing the star formation history.

`starFormationHistoryMetallicityMinimum` The upper limit to the metallicity in the lowest metallicity bin (i.e. the lowest metallicity bin will extend from zero to this value).

`starFormationHistoryMetallicityMaximum` The upper limit to the metallicity in the highest metallicity bin.

Default values set a timestep of 0.1 Gyr, with 0.01 Gyr timesteps for 0.1 Gyr before each output, along with 10 metallicity bins ranging from $10^{-4}Z_\odot$ to $10Z_\odot$. 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 §2.5). When such properties are requested, Grasil will be automatically run on each selected galaxy, the SED computed and stored to the Galacticus file[6], 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:

```
# Specify model.
my $galacticus;
$galacticus->{'file' } = "/galacticus.hdf5";
$galacticus->{'store'} = 0;
$galacticus->{'tree' } = "all";

# Specify Grasil options.
$galacticus->{'grasilOptions'}->{'includePAHs'            } = 1;
$galacticus->{'grasilOptions'}->{'fluctuatingTemperatures'} = 1;
$galacticus->{'grasilOptions'}->{'wavelengthCount'        } = 1000;
$galacticus->{'grasilOptions'}->{'radialGridCount'        } = 30;
$galacticus->{'grasilOptions'}->{'recomputeSEDs'          } = 0;
$galacticus->{'grasilOptions'}->{'launchMethod'           } = "pbs";

# Read results from model.
&HDF5::Select_Output($galacticus,2.0);
&HDF5::Get_Dataset  ($galacticus,
                     [
                      'grasilFlux850microns',
```

[6] 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.

```
 '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 tempertures 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[7].

Finally, the launchMethod option specifies how Grasil is to be run. Currently supported options are `local` and `pbs`. Selecting `local` causes Grasil to be run on the local machine. Multiple threads are launched automatically, up to one per available CPU core unless otherwise specified. Selecting `pbs` causes Grasil jobs to be submitted to a PBS queue.

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`.

4.6 Outputting Stellar Luminosities

Galacticus can compute the stellar luminosity of galaxies in any required combination of filter, redshift, and frame. To cause luminosities to be computed add something such as the following to your input parameter file:

```
<luminosityFilter   value="SDSS_r"  />
<luminosityRedshift value="0.1"     />
<luminosityType     value="observed"/>
```

This would result in a dataset called `diskLuminositiesStellar:SDSS_r:observed:z0.1000` being output (along with a similar dataset for the spheroid component), corresponding to the luminosity in the SDSS r-band filter, as observed at $z = 0.1$. To get the same filter but in the rest-frame of the galaxy, change the `luminosityType` to "`rest`". If instead of specifying a unique redshift you instead specify "`all`" for a filter in the `luminosityRedshift` element, then the filter will be replicated to all output redshifts.

Available filters can be found in the `data/filters` folder. Additionally, it is possible to specify top-hat filters (which have unit transmission over a range of wavelengths) without the need to create a filter file for them. A filter named `topHat_L_R` will be interpretted as a top-hat filter where `L`$= \lambda$ is the central wavelength (in Ångstroms) and `R`$= R$ is the resolution. A top-hat filter is constructed such that the transmission is unity between λ_1 and λ_2, and zero outside that range, and such that $\lambda_1\lambda_2 = \lambda^2$, $\lambda_2 - \lambda_1 = \lambda/R$. Furthermore, a filter specified as `topHat_Llow_Lhigh_R` will be expanded into a set of

[7] Exercise caution when using this option. Recomputing SEDs requires deleting the old SED group. The HDF5 library currently does not clean up the space occupied by the datasets in this deleted group, so the file size can grow rapidly if you repeatedly recompute Grasil SEDs.

contiguous top-hat filters with resolution `R` spanning the wavelength range `Llow` to `Lhigh` with the first filter's lower wavelenght limit aligned with `Llow` and continuing until the central wavelength of the filter is no longer below `Lhigh`.

Luminosities are always output in units of the zero-point of the AB magnitude system, such that $-2.5 \log_{10} L$ (where L is the output luminosity) gives the AB absolute magnitude of the galaxy.

You can add additional luminosities by simply adding more entries in these parameter values. For example:

```
<luminosityFilter   value="SDSS_r   SDSS_g"/>
<luminosityRedshift value="0.1      0.3"   />
<luminosityType     value="observed rest"  />
```

would result in datasets `diskLuminositiesStellar:SDSS_r:observed:z0.1000` and `diskLuminositiesStellar:SDSS_g:rest:z0.3000`.

4.6.1 Postprocessing of Stellar Spectra

Stellar luminosities are computed by convolving a library of simple stellar populations with the star formation history of each galaxy. Galacticus allows the spectra of those simple stellar populations to be postprocessed (after being read from file or internally generated for example) before they are utilized in the convolution integral. This postprocessing can modify the spectra in arbitrary ways that depend on wavelength, redshift, and age of stellar population. Furthermore, Galacticus allows you to chain together stellar spectra postprocessors into a set to allow multiple postprocessings to be applied. Furthermore again, you can define an arbitrary number of sets and apply different sets to different luminosities.

Typical uses of stellar spectra postprocessors include accounting for absorption of galaxy light by the intervening intergalactic medium (IGM), or capturing only the light from recent star formation[8]. A full list of the available postprocessors can be found in §13.49.

If you don't specify a postprocessing set, the "default" set (consisting of the `inoue2014` postprocessor; see §13.49.1) is applied to each luminosity calculation. To specify other postprocessing sets add the following to your parameter file:

```
<luminosityPostprocessSet value="default recent unabsorbed recentUnabsorbed"/>
```

where one set must be specified for each luminosity specified in the `luminosityFilter` parameter. Note that set names can be reused in order to apply the same postprocessor set to multiple luminosities.

The chain of postprocessors to apply for each set is then specified as follows:

```
<stellarPopulationSpectraPostprocessRecentMethods           value="inoue2014 recent"/>
<stellarPopulationSpectraPostprocessUnabsorbedMethods       value="identity"        />
<stellarPopulationSpectraPostprocessRecentUnabsorbedMethods value="recent"          />
```

In this case we've constructed three new sets, in addition to the default set (which applies just the `inoue2014` postprocessor). The `recent` set applies both the `inoue2014` IGM absorption postprocessor, followed by the `recent` postprocessor to retain only recently emitted light. The `unabsorbed` set ignores IGM absorption entirely—it does this by using the `identity` postprocessor which leaves the spectrum unaffected. Finally, the `recentUnabsorbed` set applies only the `recent` filter while ignoring IGM absorption.

In this way it is relatively easy to extract multiple different measures of luminosity from a Galacticus model. For example, you could construct four postprocessor sets, each corresponding to one of the four different IGM absorption models (`lycSuppress`, `madau1995`, `meiksin2006`, and `inoue2014`) and apply these to the same luminosity filter to assess how luminosity depends on the IGM model used.

[8] Perhaps so that additional dust extinction can be applied to the light of recently formed stars.

4.6.2 Migrating Parameter Files to a New Version

The names and allowed values of parameters often change between versions of Galacticus. To permit easy and error-free migration between versions a script is provided to translate parameter files from earlier to later versions. To migrate a parameter file simply use:

```
scripts/aux/parametersMigrate.pl parameters.xml newParameters.xml
```

By default, this script will translate from the previous to the current version of Galacticus. If your parameter file contains a `version` element then this will be used to determine which version of Galacticus the parameter file was constructed for. The migration script will then migrate the parameter file through all intermediate versions to bring it into compliance with the current version. You can also specify input and output versions directly:

```
scripts/aux/parametersMigrate.pl parameters.xml newParameters.xml --inputVersion 0.9.0 --outputVersion 0
```

will convert `parameters.xml` from version 0.9.0 syntax to version 0.9.3 syntax.

4.6.3 Computing Emission Lines

Galacticus can compute emission line luminosities for galaxies. This calculation is based on the methodology of Panuzzo et al. [2003]. Briefly, HII region models are constructed using Cloudy for a variety of gas densities and metallicities, and HI, HeI, and OII ionizing luminosities. Emission line luminosities are then computed by interpolating in these tables based on the instantaneous properties of model galaxies.

To compute emission line luminosities it is therefore necessary to run Galacticus including the following rest-frame luminosity filters at each redshift of interest: `Lyc`, `HeliumContinuum`, `OxygenContinuum`. This causes the ionizing luminosity for each three species to be computed and output. Emission line luminosities can then be found using Galacticus's data extraction modules (see §2.2), by simply importing the `Galacticus::EmissionLines` module. Emission line luminosites (in units of Solar luminosities) can then be accessed as named properties using names such as `totalLineLuminosity:balmerAlpha6563:rest:z0.0000`, which will return the Hα luminosity at $z = 0$. Equivalent properties for disk and spheroid are provided (simply replace "`total`" with "`disk`" or "`spheroid`"). Currently available lines are:

- `balmerAlpha6563`
- `balmerBeta4861`
- `oxygenII3726`
- `oxygenII3729`
- `oxygenIII4959`
- `oxygenIII5007`
- `nitrogenII6584`
- `sulfurII6731`
- `sulfurII6716`

Additionally, if a line is requested as `totalLineLuminosity:balmerAlpha6563:<filterName>:rest:z0.0000` then the line luminosity is computed under the provided filter, and the luminosity is returned in units of maggies for easy conversion to magnitudes.

4.6.4 Reionization Calculations

Galacticus can self-consistently solve for the evolution of the IGM as it becomes photoionized by light emitted by stars and AGN. To activate this calculation, include the following in your parameters file:

```
<!-- IGM evolver -->
<intergalacticMediumStateMethod  value="internal"/>
<igmPropertiesCompute            value="true"    />
<igmPropertiesTimeCountPerDecade value="10"      />

<!-- Background radiation -->
<backgroundRadiationCompute                   value="true"    />
<radiationIntergalacticBackgroundMethod       value="internal"/>
<backgroundRadiationWavelengthCountPerDecade value="50"      />
<backgroundRadiationTimeCountPerDecade        value="10"      />

<!-- Halo accretion options -->
<accretionHaloMethod value="naozBarkana2007"/>
```

The first block of parameters switches Galacticus to using an internal calculation for the state of the IGM, instructs it to solve for IGM properties as a function of time, and specifies that IGM properties should be updated 10 times per decade of cosmic time. Specifically, at each of these time intervals, solving of galaxy evolution is halted and the IGM evolved up to this time using the currently computed photoionizing background spectrum.

The second block of parameters activates an internal calculation of cosmic background radiation, in which the background is computed from the emissivities of model galaxies and AGN. The number of points at which to tabulate the background per decade of wavelength and cosmic time are specified.

Finally, the third block of parameters tells Galacticus to use the Naoz and Barkana [2007] prescription for computing gas accretion into halos from the IGM. This prescription uses the filtering mass to determine accretion rates, and will take the filtering mass from the internal IGM evolution calculation.

With these three sets of configurations, Galacticus will perform a self-consistent evolution of the IGM—in the sense that the IGM is ionized by photons emitted by model galaxies and AGN, while galaxy evolution is affected by the computed state of the model IGM. Note that, when run in this way, Galacticus needs to keep all merger trees in memory simultaneously (as they are run synchronously to allow the IGM properties to evolved alongside galaxy properties).

Once completed, data on the IGM and background radiation are written to the output file in the `igmProperties` and `backgroundRadiation` groups respectively.

Bibliography

Michael Boylan-Kolchin, Chung-Pei Ma, and Eliot Quataert. Dynamical friction and galaxy merging timescales. *MNRAS*, 383:93–101, 2008. URL http://adsabs.harvard.edu/abs/2008MNRAS.383...93B. 4.1.1

Stéphane Charlot and S. Michael Fall. A simple model for the absorption of starlight by dust in galaxies. *The Astrophysical Journal*, 539:718–731, August 2000. URL http://adsabs.harvard.edu/abs/2000ApJ...539..718C. 2.2.1

Andrea Ferrara, Simone Bianchi, Andrea Cimatti, and Carlo Giovanardi. An atlas of monte carlo models of dust extinction in galaxies for cosmological applications. *Astrophysical Journal Supplement Series*, 123:437–445, August 1999. URL http://adsabs.harvard.edu/abs/1999ApJS..123..437F. 2.2.1, 4.4

Stuart P. D. Gill, Alexander Knebe, and Brad K. Gibson. The evolution of substructure - i. a new identification method. *Monthly Notices of the Royal Astronomical Society*, 351:399–409, June 2004. ISSN 0035-8711. doi: 10.1111/j.1365-2966.2004.07786.x. URL http://adsabs.harvard.edu/abs/2004MNRAS.351..399G. 4.6.4

Martha P. Haynes, Riccardo Giovanelli, Ann M. Martin, Kelley M. Hess, Amélie Saintonge, Elizabeth A. K. Adams, Gregory Hallenbeck, G. Lyle Hoffman, Shan Huang, Brian R. Kent, Rebecca A. Koopmann, Emmanouil Papastergis, Sabrina Stierwalt, Thomas J. Balonek, David W. Craig, Sarah J. U. Higdon, David A. Kornreich, Jeffrey R. Miller, Aileen A. O'Donoghue, Ronald P. Olowin, Jessica L. Rosenberg, Kristine Spekkens, Parker Troischt, and Eric M. Wilcots. The arecibo legacy fast ALFA survey: The Îś.40 h i source catalog, its characteristics and their impact on the derivation of the h i mass function. *The Astronomical Journal*, 142:170, November 2011. ISSN 0004-6256. doi: 10.1088/0004-6256/142/5/170;. URL http://adsabs.harvard.edu/abs/2011AJ....142..170H. 2.2, 2.8.1

Christopher C. Hayward, DuÅąan KereÅą, Patrik Jonsson, Desika Narayanan, T. J. Cox, and Lars Hernquist. What does a submillimeter galaxy selection actually select? the dependence of submillimeter flux density on star formation rate and dust mass, December 2010. URL http://adsabs.harvard.edu/abs/2011arXiv1101.0002H. 2.2.1, 2.2.1

Philip F. Hopkins, Gordon T. Richards, and Lars Hernquist. An observational determination of the bolometric quasar luminosity function. *The Astrophysical Journal*, 654:731–753, January 2007. URL http://adsabs.harvard.edu/abs/2007ApJ...654..731H. 2.2.1

Steffen R. Knollmann and Alexander Knebe. AHF: amiga's halo finder. *The Astrophysical Journal Supplement Series*, 182:608–624, June 2009. ISSN 0067-0049. doi: 10.1088/0067-0049/182/2/608. URL http://adsabs.harvard.edu/abs/2009ApJS..182..608K. 4.6.4

Ann M. Martin, Emmanouil Papastergis, Riccardo Giovanelli, Martha P. Haynes, Christopher M. Springob, and Sabrina Stierwalt. The arecibo legacy fast ALFA survey. x. the h i mass function and Îľ_H i from the 40% ALFALFA survey. *The Astrophysical Journal*, 723:1359–1374, November 2010. ISSN 0004-637X. doi: 10.1088/0004-637X/723/2/1359;. URL http://adsabs.harvard.edu/abs/2010ApJ...723.1359M. 2.8.1

S. Naoz and R. Barkana. The formation and gas content of high-redshift galaxies and minihaloes. *Monthly Notices of the Royal Astronomical Society*, 377:667–676, May 2007. ISSN 0035-8711. doi: 10.1111/j.1365-2966.2007.11636.x. URL http://adsabs.harvard.edu/abs/2007MNRAS.377..667N. 4.6.4

R. Narayan, R. Mahadevan, and E. Quataert. Advection-dominated accretion around black holess. page 148, 1998. URL http://adsabs.harvard.edu/abs/1998tbha.conf..148N. 4.6.4

D. Obreschkow, D. Croton, G. De Lucia, S. Khochfar, and S. Rawlings. Simulation of the cosmic evolution of atomic and molecular hydrogen in galaxies. *The Astrophysical Journal*, 698:1467–1484, June 2009. ISSN 0004-637X. doi: 10.1088/0004-637X/698/2/1467. URL http://adsabs.harvard.edu/abs/2009ApJ...698.1467O. 2.8.1, 2.8.1

P. Panuzzo, A. Bressan, G. L. Granato, L. Silva, and L. Danese. Dust and nebular emission. i. models for normal galaxies. *Astronomy and Astrophysics*, 409:99–114, October 2003. URL http://adsabs.harvard.edu/abs/2003%26A...409...99P. 4.6.3

Laura Silva, Gian Luigi Granato, Alessandro Bressan, and Luigi Danese. Modeling the effects of dust on galactic spectral energy distributions from the ultraviolet to the millimeter band. *The Astrophysical Journal*, 509:103–117, December 1998. URL http://adsabs.harvard.edu/abs/1998ApJ...509..103S. 2.5

Ryuichi Takahashi, Masamune Oguri, Masanori Sato, and Takashi Hamana. Probability distribution functions of cosmological lensing: Convergence, shear, and magnification. *The Astrophysical Journal*, 742:15, November 2011. doi: 10.1088/0004-637X/742/1/15;. URL http://adsabs.harvard.edu/abs/2011ApJ...742...15T. 2.2.1

M. A. Zwaan, M. J. Meyer, L. Staveley-Smith, and R. L. Webster. The HIPASS catalogue: $\omega_{\rm HI}$ and environmental effects on the HI mass function of galaxies. *Monthly Notices of the Royal Astronomical Society*, 359:L30–L34, May 2005. URL http://adsabs.harvard.edu/abs/2005MNRAS.359L..30Z. 2.8.1

Martin A. Zwaan, Frank H. Briggs, David Sprayberry, and Ertu Sorar. The h i mass function of galaxies from a deep survey in the 21 centimeter line. *The Astrophysical Journal*, 490:173, November 1997. ISSN 0004-637X. doi: 10.1086/304872. URL http://adsabs.harvard.edu/abs/1997ApJ...490..173Z. 2.8.1

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. 48

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). 49

AHF Amiga's Halo Finder (AHF) is a software package which identifies dark matter halos in N-body simulations. Full details are given by Gill et al. [2004] and Knollmann and Knebe [2009]. 49

BIE The Bayesian Inference Engine (BIE) is a software package designed to facilitate exploration of complexes parameter spaces using Bayesian techniques.. 49

CDF Cumulative Distribution Function (CDF) is a function which describes the cumulative probability for a random variable to be equal to or less than a given value.. 49

CDM Cold dark matter (CDM) is a hypothesized type of dark matter in which the particles move slowly compared to the speed of light. 49

component An individual physical system within a node, such as a dark matter halo, a galactic disk or a supermassive black hole. 47

deadlock A deadlock describes a situation in which no node in a merger tree (or forest) can be evolved further forward in time due to the existence of circular dependencies between nodes. Deadlocks can occur due to incompatible parameter choices, or may indicate a bug in Galacticus.. 28

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. 49

forest A collection of merger trees that are linked together by virtue of nodes which jump between trees. 28

GAMA The Galaxy and Mass Assembly (GAMA) survey is a spectroscopic survey of $\approx 300,000$ galaxies down to r < 19.8 mag over ≈ 286 deg^2.. 49

GSL GNU Scientific Library (GSL) is a library providing a variety of numerical algorithms.. 49

HDF5 The hierarhical data format (version 5; HDF5) is a file format designed for storing scientific data.. 49

HOD A halo occupation distribution (HOD) is a mathematical model describing the distribution of the number of galaxies (of some given physical properties) found in a dark matter halo of given mass.. 49

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. 12

maggie A unit of luminosity defined to be equal to the luminosity of a zeroth magnitude object in the AB magnitude system. 13, 42

MPI Message Passing Interface (MPI) is a standard for passing messages between processes on parallel computers.. 49

node A single point in a merger tree, consisting of a dark matter halo and associated baryons. 28, 29, 47

PAH Polycyclic aromatic hydrocarbons (PAH) are large organic molecules consisting of fused aromatic rings. 49

PBS Portable Batch System (PBS) is a job scheduler used on many compute cluster environments.. 49

PDF Probability Density Function (PDF) is a function which describes the probability for a random variable to take on a given value.. 49

SAM Semi-analytic models (SAMs) are a type of galaxy formation model utilizing simple parameterizations of physical processes to follow the evolution of galaxies through a merging hierarchy of galaxies.. 49

UUID A universally unique identifier—this is a label which uniquely identifies some object (in this case, a Galacticus model). 17

WDM Warm dark matter (WDM) is a hypothesized type of dark matter in which the particle has non-negligible thermal velocity at decoupling. 49

Acronyms

ADAF advection-dominated accretion flow. *Glossary:* ADAF

AHF Amiga's Halo Finder. *Glossary:* AHF

BIE semi-analytic model. *Glossary:* BIE

CDF cumulative distribution function. *Glossary:* CDF

CDM cold dark matter. *Glossary:* CDM

DSL domain-specific language. *Glossary:* DSL

GAMA Galaxy and Mass Assembly. *Glossary:* GAMA

GSL GNU Scientific Library. *Glossary:* GSL

HDF5 hierarchical data format. 11, 38, *Glossary:* HDF5

HOD halo occupation distribution. *Glossary:* HOD

IGM intergalactic medium. 41, 43

IMF initial mass function. 37

ISM interstellar medium. 20

MPI message passing interface. *Glossary:* MPI

PAH polycyclic aromatic hydrocarbon. 40, *Glossary:* PAH

PBS portable batch system. *Glossary:* PBS

PDF probability density function. *Glossary:* PDF

SAM semi-analytic model. *Glossary:* SAM

SED spectral energy distribution. 17, 39, 40

SNe supernovae. 37

WDM warm dark matter. *Glossary:* WDM

Index