astro3D Documentation

Jacob Seiler, Manodeep Sinha

Nov 05, 2018

Contents

1 Utilities 1

2 astro3D package 5

Python Module Index 11

CHAPTER 1

Utilities

This directory contains a number of useful tools and utilities for handling the data from the ASTRO3D Genesis simulations. We have provided a number of example scripts to run these tools in the **run_scripts** directory.

1.1 adjust_spec

This utility adjusts the data specification of the default `VELOCIraptor``+``Wherewolf` trees to match the required specification of the LHaloTree strucutre. See LHaloTreeReader for an overview of the LHalo tree structure.

1.2 forest_sorter

This utility takes input HDF5 merger trees that have not been saved in any specific order and sorts them on a number of fields; usually an at least an ID field and a mass field. The user can specify on which fields the sorting should occur. In the default case, the trees are sorted first on the `ForestID`, then the `hostHaloID` and finally in descending order of 200mean mass of the halo. This results in halos that are sorted in ascending order according to their `ForestID`, then within each Forest, all halos within a single FoF group are grouped and finally within each FoF-group, the most massive halo is sorted first.

1.2.1 Tests

First please run the basic tests on the default test data provided by invoking `pytest`. If this default test does not pass, please email jseiler@swin.edu.au

Included in the main function of `tests/forest_sorter_test.py` is an example of running tests with customized settings. Of particular note is the `gen_data` variable. If this is set to `1`, a small set of sorted trees will be generated from the specified unsorted HDF5 trees. The number of halos tested on is handled by the `NHalos_test` variable.

If you wish to test your fully sorted trees after running `forest_sorter()`, set `gen_data=0`, `fname_in` to the path of the **original unsorted** trees and `fname_out` to the path of the **sorted** trees.

If the default test passes but your specific test fails please ensure that your data file is not corrupt. Importantly, check that the snapshot keys are named appropriately. We require the snapshot fields to include the word **snap** (case insensitive) and assume that the snapshot number corresponding to the snapshot key is included as a single cluster towards the end of the key; **snap53_04** should correspond to snapshot number 04 for example.

If the snapshot fields are named correctly and your data can be otherwise read in via 'h5py', please email jseiler@swin.edu.au

1.3 convert_indices

This utility takes the sorted HDF5 trees from `forest_sorter` and adjusts the halo IDs to match the requirements of LHalo trees. LHalo trees requires that these IDs are tree local and are the indices of the halos (rather than unique temporal IDs).

The resulting LHalo compatible trees are saved as a HDF5 file with all other fields identical to the input trees.

1.4 treefrog_to_lhalo

This function takes trees with the LHalo corrected indices (from `convert_indices()`) and writes LHalo tree format. Under this format, the tree pointers used to walk the tree are **tree local**. See LHaloTreeReader for an overview of the LHalo tree style pointers.

The output format for this function can be either binary (`write_binary_flag == 1`), HDF5 (`write_binary_flag == 0`) or both (`write_binary_flag == 2`).

The binary files will have the following data format:

- 32-bit integer: *NTrees*, describing the number of trees in the file,
- 32-bit integer: *TotNHalos*, describing the total number of halos within the file,
- *NTrees* 32-bit integers: *TreeNHalos*, describing the number of halos within each tree.

Following this header is *TotNHalos* halo entries with data format:

- *Descendant*, 32-bit integer,
- *FirstProgenitor*, 32-bit integer,
- *NextProgenitor*, 32-bit integer,
- *FirstHaloInFOFgroup*, 32-bit integer,
- *NextHaloInFOFgroup*, 32-bit integer,
- *Len*, 32-bit integer,
- *M_Mean200*, 32-bit float,
- *Mvir*, 32-bit float,
- *M_TopHat*, 32-bit float,
- *Posx*, 32-bit float,
- *Posy*, 32-bit float,
- *Posz*, 32-bit float,
- *Velx*, 32-bit float,
- *Vely*, 32-bit float,

- *Velz*, 32-bit float,
- *VelDisp*, 32-bit float,
- *Vmax*, 32-bit float,
- *Spinx*, 32-bit float,
- *Spiny*, 32-bit float,
- *Spinz*, 32-bit float,
- *MostBoundID*, 64-bit integer,
- *SnapNum*, 32-bit integer,
- *Filenr*, 32-bit integer,
- *SubHaloIndex*, 32-bit integer,
- *SubHalfMass*, 32-bit integer.

The function is MPI compatible and the final number of files written is equivalent to the number of processors used to call the function. These files are load balanced such that each one will have a similar number of halos (but not necessarily number of trees). For example,

```
$ mpirun -np 4 python run_scripts/run_treefrog_to_lhalo.py
```

Would generate 4 LHalo tree binary files.

CHAPTER 2

astro3D package

2.1 Subpackages

2.1.1 astro3D.genesis package

Subpackages

astro3D.genesis.utils package

Submodules

astro3D.genesis.utils.adjust_spec module

Authors: Jacob Seiler, Manodeep Sinha

`astro3D.genesis.utils.adjust_spec.`**`adjust_spec`**(*fname_in*, *fname_out*, *haloID_field='ID'*, *FirstHaloInFOFgroup_field='hostHaloID'*, *index_mult_factor=1000000000000*)

Adjusts some fields of the VELOCIraptor trees to match the LHaloTree Specs.

Currently calls the following functions:

1. `astro3D.genesis.utils.`adjust_hostHaloID

Parameters

- **fname_in, fname_out** (*String*) – Path to the input HDF5 trees and path to where the updated trees will be saved.
- **haloID_field** (*String, optional*) – Field name within the HDF5 file that corresponds to the unique halo ID.

- **FirstHaloInFOFgroup_field** (*String, optional*) – Field name within the HDF5 file that corresponds to *FirstHaloInFOFgroup* in the LHaloTree structure.
- **index_mult_factor** (*Integer, optional*) – Multiplication factor to generate a temporally unique halo ID.

Returns

Return type None.

Notes

The default parameters are chosen to match the ASTRO3D Genesis trees as produced by VELOCIraptor + Treefrog.

astro3D.genesis.utils.adjust_spec.**adjust_hostHaloID**(*f_out*, *haloID_field*, *FirstHaloInFOFgroup_field*, *Snap_Keys*, *Snap_Nums*, *index_mult_factor*)

Adjusts the *hostHaloID* field in the output HDF5 file.

In the original trees, if a halo is the main background FoF halo, its value of *hostHaloID* is set to *-1*. Under the LHaloTree specs, the property correpsonding to this field (*FirstHaloInFOFgroup*) can never be -1. Instead, halos in these instances should point to themselves.

Parameters

- **f_out** (*Open HDF5 file.*) – The HDF5 trees we're adjusting.
- **haloID_field** (*String, optional*) – Field name within the HDF5 file that corresponds to the unique halo ID.
- **FirstHaloInFOFgroup_field** (*String, optional*) – Field name within the HDF5 file that corresponds to *FirstHaloInFOFgroup* in the LHaloTree structure.
- **Snap_Keys** (*List of strings.*) – Names of the snapshot keys within the passed keys.
- **Snap_Nums** (Dictionary of integers keyed by *Snap_Keys*.) – Snapshot number of each snapshot key.
- **index_mult_factor** (*Integer, optional*) – Multiplication factor to generate a temporally unique halo ID.

Returns

Return type None.

astro3D.genesis.utils.common module

astro3D.genesis.utils.convert_indices module

astro3D.genesis.utils.convert_indices.**convert_indices**(*fname_in*, *fname_out*, *haloID_field='ID'*, *forestID_field='ForestID'*, *ID_fields=['Head', 'Tail', 'RootHead', 'RootTail', 'ID', 'hostHaloID']*, *index_mult_factor=1000000000000*)

Converts temporally unique tree IDs to ones that are forest-local as required by the LHalo Trees format.

The data-structure of the Treefrog trees is assumed to be HDF5 File -> Snapshots -> Halo Properties at each snapshot.

A new HDF5 file is saved out with the updated IDs.

Note: We require the input trees to be sorted via the forest ID (`forestID_field`) and suggest to also sub-sort on `hostHaloID` and mass. Sorting can be done using *astro3D.genesis.utils.forest_sorter*.

Parameters

- **fname_in, fname_out** (*String*) – Path to the input HDF5 VELOCIraptor + treefrog trees and the path where the LHalo correct trees will be saved.
- **haloID_field** (*String, optional*) – Field name within the HDF5 file that corresponds to the unique halo ID.
- **forestID_field** (*String, optional*) – Field name within the HDF5 file that corresponds to forest ID.
- **ID_fields** (*List of strings, optional*) – The HDF5 field names that correspond to properties that use halo IDs. As the halo IDs are updated to match the required LHalo Tree format, these must also be updated.
- **index_mult_factor** (*Integer, optional*) – Multiplication factor to generate a temporally unique halo ID.

Returns

Return type None.

Notes

The default parameters are chosen to match the ASTRO3D Genesis trees as produced by VELOCIraptor + Treefrog.

astro3D.genesis.utils.forest_sorter module

Authors: Jacob Seiler, Manodeep Sinha

`astro3D.genesis.utils.forest_sorter.`**`forest_sorter`**(*fname_in, fname_out, haloID_field='ID', sort_fields=['ForestID', 'hostHaloID', 'Mass_200mean'], sort_direction=[1, 1, -1], ID_fields=['Head', 'Tail', 'RootHead', 'RootTail', 'ID', 'hostHaloID'], index_mult_factor=1000000000000*)

Sorts and saves a HDF5 tree file on the specified sort fields. The IDs of the halos are assume to use the index within the data file and hence will be updated to reflect the sorted order.

Parameters

- **fname_in, fname_out** (*String*) – Path to the input HDF5 trees and path to where the sorted trees will be saved.

- **haloID_field** (*String, optional*) – Field name within the HDF5 file that corresponds to the unique halo ID.
- **sort_fields** (*List of strings, optional*) – The HDF5 field names that the sorting will be performed on. The entries are ordered such that the first field will be the outer-most sort and the last field will be the inner-most sort.
- **sort_direction** (*List of integers, optional*) – Specifies the direction in which the sorting will occur for each `sort_field` entry. 1 corresponds to ascending, -1 to descending.
- **ID_fields** (*List of strings, optional*) – The HDF5 field names that correspond to properties that use halo IDs. As the halo IDs are updated to reflect the new sort order, these fields must also be updated.
- **index_mult_factor** (*Integer, optional*) – Multiplication factor to generate a temporally unique halo ID.

Returns

Return type None.

Notes

The default parameters are chosen to match the ASTRO3D Genesis trees as produced by VELOCIraptor + Treefrog.

astro3D.genesis.utils.treefrog_to_lhalo module

Authors: Jacob Seiler, Manodeep Sinha

`astro3D.genesis.utils.treefrog_to_lhalo.`**`treefrog_to_lhalo`**(*fname_in*, *fname_out*, *haloID_field='ID'*, *forestID_field='ForestID'*, *Nforests=None*, *write_binary_flag=1*, *debug=0*)

Takes the Treefrog trees that have had their IDs corrected to be in LHalo format and saves them in LHalo binary format.

The data-structure of the Treefrog trees is assumed to be HDF5 File -> Snapshots -> Halo Properties at each snapshot.

Note: We require the input trees to be sorted via the forest ID (`forestID_field`) and suggest to also subsort on hostHaloID and mass. Sorting can be done using *astro3D.genesis.utils.forest_sorter*.

We also require the input trees to have IDs that are LHalo compatible. See *astro3D.genesis.utils.convert_indices*.

Parameters

- **fname_in, fname_out** (*String*) – Path to the input HDF5 VELOCIraptor + treefrog trees and the path where the LHalo binary file will be saved.
- **haloID_field** (*String, optional*) – Field name within the HDF5 file that corresponds to the unique halo ID.

- **forestID_field** (*String, optional*) – Field name within the HDF5 file that corresponds to forest ID.
- **Nforests** (*Integer, optional*) – The number of forests to be processed. If `None` is passed then all forests are processed.
- **write_binary_flag** (*Integer, optional*) – Flag to decide whether to write to a binary or HDF5 file. 0: HDF5 file only. 1: Binary file only. 2: Both binary and HDF5 file.

Returns

Return type None.

Notes

The default parameters are chosen to match the ASTRO3D Genesis trees as produced by VELOCIraptor + Treefrog.

Python Module Index

a

astro3D, 5
astro3D.genesis, 5
astro3D.genesis.utils, 5
astro3D.genesis.utils.adjust_spec, 5
astro3D.genesis.utils.common, 6
astro3D.genesis.utils.convert_indices, 6
astro3D.genesis.utils.forest_sorter, 7
astro3D.genesis.utils.treefrog_to_lhalo, 8

Index

A

adjust_hostHaloID() (in module astro3D.genesis.utils.adjust_spec), 6
adjust_spec() (in module astro3D.genesis.utils.adjust_spec), 5
astro3D (module), 5
astro3D.genesis (module), 5
astro3D.genesis.utils (module), 5
astro3D.genesis.utils.adjust_spec (module), 5
astro3D.genesis.utils.common (module), 6
astro3D.genesis.utils.convert_indices (module), 6
astro3D.genesis.utils.forest_sorter (module), 7
astro3D.genesis.utils.treefrog_to_lhalo (module), 8

C

convert_indices() (in module astro3D.genesis.utils.convert_indices), 6

F

forest_sorter() (in module astro3D.genesis.utils.forest_sorter), 7

T

treefrog_to_lhalo() (in module astro3D.genesis.utils.treefrog_to_lhalo), 8