VELOCIraptor Documentation

Release 1.60

Pascal Jahan Elahi, Rhys Poulton, Rodrigo Canas

Nov 01, 2021

CONTENTS:

VELOCIraptor is a C++ halo finder using MPI and OpenMP APIs. The repository also contains several associated analysis tools in python, example configuration files and analysis python scripts (and sample jupyter notebooks). The code can also be compiled as a library for on-the-fly halo finding within an N-body/hydrodynamnical code. Currently integration is limited to SWIFTSIM but extensions are in the works for other codes.

There is an associated halo merger tree code TreeFrog (also C++ MPI+OpenMP).

If you are using **VELOCIraptor** please cite the following paper, which describe the code in full:

```
@ARTICLE{doi:10.1017/pasa.2019.12,
  author = {{Elahi}, Pascal J. and {Ca{\~n}as}, Rodrigo and {Poulton}, Rhys J.~J. and
↪{Tobar}, Rodrigo J. and {Willis}, James S. and {Lagos}, Claudia del P. and {Power},␣
↪Chris and {Robotham}, Aaron S.~G.},
  title = {Hunting for galaxies and halos in simulations with VELOCIraptor},
  journal = {\pasa},
  keywords = {dark matter, galaxies: evolution, galaxies: halos, methods: numerical,␣
↪Astrophysics -Cosmology and Nongalactic Astrophysics},
  year = {2019},
  month = {Jan},
  volume = {36},
  eid = {e021},
  pages = {e021},
  doi = {10.1017/pasa.2019.12},
  archivePrefix = {arXiv},
  eprint = {1902.01010},
  adsurl = {https://ui.adsabs.harvard.edu/abs/2019PASA...36...21E},
}
```

If using **VELOCIraptor** for galaxy finding, please also cite:

```
@ARTICLE{doi:10.1093/mnras/sty2725,
  author = {{Ca{\~n}as}, Rodrigo and {Elahi}, Pascal J. and {Welker}, Charlotte and {del␣
↪P Lagos}, Claudia and {Power}, Chris and {Dubois}, Yohan and {Pichon}, Christophe},
  title = {Introducing a new, robust galaxy-finder algorithm for simulations},
  journal = {\mnras},
  keywords = {methods: numerical, galaxies: evolution, dark matter, cosmology: theory,␣
↪Astrophysics - Astrophysics of Galaxies},
  year = {2019},
  month = {Jan},
  volume = {482},
  number = {2},
  pages = {2039-2064},
  doi = {10.1093/mnras/sty2725},
  archivePrefix = {arXiv},
```

(continues on next page)

(continued from previous page)

```
  eprint = {1806.11417},
  primaryClass = {astro-ph.GA},
  adsurl = {https://ui.adsabs.harvard.edu/abs/2019MNRAS.482.2039C},
}
```

The original idea, which also discusses the identification of tidal debris is in:

```
@ARTICLE{doi:10.1111/j.1365-2966.2011.19485.x,
  author = {{Elahi}, Pascal J. and {Thacker}, Robert J. and {Widrow}, Lawrence M.},
  title = {Peaks above the Maxwellian Sea: a new approach to finding substructures in N-
↪body haloes},
  journal = {\mnras},
  keywords = {methods: data analysis, methods: numerical, galaxies: haloes, galaxies:␣
↪structure, dark matter, Astrophysics - Cosmology and Extragalactic Astrophysics},
  year = {2011},
  month = {Nov},
  volume = {418},
  number = {1},
  pages = {320-335},
  doi = {10.1111/j.1365-2966.2011.19485.x},
  archivePrefix = {arXiv},
  eprint = {1107.4289},
  primaryClass = {astro-ph.CO},
  adsurl = {https://ui.adsabs.harvard.edu/abs/2011MNRAS.418..320E},
}
```

An online entry can also be found at NASA's ADS service.

CHAPTER

ONE

GETTING VELOCIRAPTOR

VELOCIraptor is currently hosted in GitHub. To get a copy you can clone the repository:

```
git clone https://github.com/pelahi/VELOCIraptor-STF
```

VELOCIraptor's compilation system is based on cmake. `cmake` will check that you have a proper compiler (anything supporting C++11 or later should do), and scan the system for all required dependencies.

To compile **VELOCIraptor** run (assuming you are inside the `VELOCIraptor-STF/` directory already):

```
$> mkdir build
$> cd build
$> cmake ..
$> make
```

With `cmake` you can also specify additional compilation flags. For example, if you want to generate the fastest possible code you can try this:

```
$> cmake .. -DCMAKE_CXX_FLAGS="-O3 -march=native"
```

You can also specify a different installation directory like this:

```
$> cmake .. -DCMAKE_INSTALL_PREFIX=~/my/installation/directory
```

Other `cmake` options that can be given in the command-line include:

A list of compile time options is found below in *Compilation Options*.

1.1 Requirments

VELOCIraptor depends on:

- GSL - the GNU Scientific Library
- **NBodylib** - a internal scientific library included with **VELOCIraptor**. **VELOCIraptor** needs this library for a number of structures, classes, and methods it provides.

1.1.1 Optional requirements

For parallel use may need the following libraries are required for compilation depending on the compilation flags used:

- MPI - the Message Passing Interface (version 1.0 or higher). Many vendor supplied versions exist, in addition to excellent open source implementations, e.g. Open MPI, MPICH or LAM.
- OpenMP - API, generally included with many compilers

VELOCIraptor also can output in a variety of formats: ASCII, binary, HDF and ADIOS. HDF and ADIOS can be enabled and disabled, and require libraries.

- Hiearchical Data Format (HDF) - self describing data format.
- Adaptable IO System (ADIOS) - self describing data format.

1.2 Compilation Options

These can be passed to `cmake`

External library flags

- **Parallel APIs can be enabled by setting**
  - **For MPI**

    `VR_MPI`: boolean to compile with MPI support

    `VR_MPI_REDUCE`: boolean that reduces impact of MPI memory overhead at the cost of extra cpu cycles. Suggested this be turned on

    `MPI_LIBRARY`: specify library path to MPI

    `MPI_EXTRA_LIBRARY`: Extra MPI libraries to link against

    `VR_LARGE_MPI_DOMAIN` : Enable if mpi domain is going to contain more than max 16 bit integer number of mpi processes
  - **For OpenMP**

    `NBODY_OPENMP`: boolean to compile with OpenMP support

    `OpenMP_CXX_FLAGS`: string, compiler flag that enables OpenMP
- **Enable input/output formats**
  - **For HDF**

    `VR_HDF5`: boolean on whether to include HDF support

    `VR_ALLOWPARALLELHDF5`: boolean on whether to allow for parallel HDF support (if available)

    `VR_ALLOWPARALLELHDF5COMPRESSIONHDF5`: boolean on whether to allow for compression parallel HDF support (THIS IS UNSTABLE, USE WITH CAUTION)

    `HDF5_ROOT`: specify a local directory containing HDF library.
  - **for XDR (nchilada) input**

    `VR_XDR`: boolean on whether to include XDR support

    `XDR_DIR`: specify a local directory containing XDR library.
  - **for adios output (alpha, not yet available)**

    `VR_ADIOS`: boolean on whether to include ADIOS support

ADIOS_DIR: specify a local directory containing ADIOS library.

- **To set directories of required libraries**
  - **Set the directories of the following libraries**

    GSL_DIR =

Internal precision and data structure flags

- **Adjust precision in stored variables and calculations**
  - **Calculations/properties at 32 bit float precision** VR_SINGLE_PRECISION=ON
  - **all integers are 64 bit integers. Enable this if dealing with more than MAXINT total number of particles**
    VR_LONG_INT=ON
- **Adjust NBodylib Particle class data precision and memory footprint**
  - **Do not store the mass as all particles are the same mass. WARNING:** ***This is not fully implemented for all types of***
    VR_NO_MASS=ON
  - **Use single precision to store positions,velocities, and possibly other internal properties**
    NBODY_SINGLE_PARTICLE_PRECISION=ON
  - **Use unsigned ints (size set by whether using long int or not) to store permanent 'particle' ids**
    NBODY_UNSIGNED_PARTICLE_PIDS=ON
  - **Use unsigned ints (size set by whether using long int or not) to store ids (index value). Note that velociraptor use**
    NBODY_UNSIGNED_PARTICLE_IDS=ON
- **Hydro simulations: activate extra data structures in the NBodylib Particle class**
  - **activate gas, store self-energy** VR_USE_GAS=ON
  - **activate stars only, store metallicity, formation time, star foramtion rate (for gas particles)**
    VR_USE_STARS=ON
  - **Calculate bulk black hole properties** VR_USE_BH=ON
  - **stars and gas and black holes** VR_USE_HYDRO=ON
- **Adjust memory/max size of Binary KD Tree options, used to run search particles. If tree is going to be built on more than**
  VR_USE_LARGE_KDTREE=ON

Operation flags

- **only calculate local density distribution for particles residing in field objects (but using all particles to estimate quantity).**
  VR_STRUCTURE_DEN=ON
- **or just use particles inside field objects, reducing cpu cycles but will bias estimates for particle in outer region of field stru**
  VR_HALO_DEN=ON
- **flag useful for zoom simulations with a high resolution region** VR_ZOOM_SIM=ON

Executable flags

- **Produce SWIFTSIM compatible library (executable still produced but does simply returns warning)**

```
        VR_USE_SWIFT_INTERFACE=ON
        CMAKE_CXX_FLAGS=-fPIC
```

- **Enable debugging** `DEBUG=ON`

CHAPTER

TWO

USING VELOCIRAPTOR

2.1 Running the code

Velociraptor is a stand alone executable (with the executable named stf (or STructure Finder for historical reasons)). It can be run in serial, with OpenMP, or MPI APIs. A typical command to start the code looks like:

```
./stf < args >
```

When compiled with OpenMP, setting the environment variable `OMP_NUM_THREADS` will set the number of threads in the openmp sections.

With MPI using 8 MPI threads:

```
mpirun -np 8 ./stf < args >
```

where here we assume that the parallel environment uses the `mpirun` command to start MPI applications. Depending on the operating system, other commands may be required for this task, e.g. `srun` on some Cray machines. Note that the code can in principle be started using an arbitrary number of mpi threads, but the mpi decomposition is most efficient for powers of 2.

The output produced by VELOCIraptor will typically consist of several files containing: bulk properties of structures found; particles belonging to these structures; and several additional files containing configuration information.

When running in MPI, currently each mpi thread writes its own output unless the code has been compiled with a parallel HDF5 library and HDF5 output is requested. In that case, a single file is written containing data from all threads for each type of output requested.

Note: At the moment, mpirun assumes that a single structure can fit onto the memory local to the mpi thread. If larger field objects (haloes) are to beanalyzed such that they are unlikely to fit into local memory, it is suggested another machine be used. Revision is in the works to use the Singlehalo_search option after field halos have been identified.

2.1.1 Arguments

The code has several command line arguements. To list the arguments, type

```
./stf -?
```

The arguments that can be passed are:

-i < file name of input file >

-s < number of files over which input is split >

-I < input format [1 Gadget, 2 HDF5, 3 Tipsy, 4 RAMSES, 5 NCHILADA] >

-Z < number of files to read in parallel (when mpi is invoked) >

-o < output base name (this can be overwritten by a configuration option in the config file. Suggestion would be to not use this option in the config file, use explicit command>

-C < configuration file name (see *Configuration File*) >

Of these arguements, only an input file and an output name must be provided. In such a case, it is assumed that there is only 1 input file, 1 read mpi thread, and default values for all other confirguration options. We suggest you do NOT run the code in this fashion. Instead we suggest the code be run with at least a configuration file passed.

```
./stf -i input -o output -C configfile.txt
```

This configuration file is an ascii file that lists keywords and values. A list of keywords, along with a description is presented below in *Configuration File*. A more typical command for a large cosmological simulation might be something like

```
export OMP_NUM_THREADS=4
mpirun -np 64 ./stf -i somehdfbasename -s 128 -I 2 -Z 64 -o output -C configfile.txt >␣
↪stf.log
```

2.1.2 Running within swiftsim

VELOCIraptor is also able to be called from within an N-body/Hydrodynamnical code as a library. Currently the code has been integrated in to **swifsim**. Details can be found in the **swiftsim** documentation. The key is that the **swiftsim** code's configuration file lists the **VELOCIraptor** configuration file used to run **VELOCIraptor**.

2.1.3 Output

Here we provide a *brief* description of the standard data products provided by **VELOCIraptor**. For a more detailed discussion and some sample analysis using these data products see *Understanding and Analysing VELOCIraptor Output*.

When operating in a typical configuration with typical compile time options, the executable (or each mpi thread) will produce several files (with the mpi threads appending their rank to the end of the file name, unless parallel HDF5 output is requested). The files typically produced are :

Output files

- `.properties`: a file containing the bulk properties of all structures identified.
- `.catalog_groups`: a file containing the size of the structures (in number of particles associated) & information need to read particle information produced by velociraptor

- `.catalog_particles`: a file containing a list of particle IDs of those in structures. Information contained in `.catalog_groups` is used to parse this data.
- `.catalog_particles.unbound`: similar to `catalog_particles` but lists particles in structures but are formally unbound. Information contained in `.catalog_groups` is used to parse this data.

Extra output files

- `.profiles`: a file containing the radial mass profiles of (sub)halos
- `.catalog_parttypes`: a file similar to `.catalog_particles` but stores particle type instead of paricle id.
- `.catalog_parttypes.unbound`: a file similar to `.catalog_parttypes` but for unbound particles.
- `.extendedinfo`: a file containing extra information on where particles are located in the input file for quick extraction from said input file of particles within groups. Still in alpha
- `.catalog_SOlist`: a file containing particle IDs within the spherical overdensity region of halos.

2.1.4 Configuration File

An example configuration file can be found the examples directory within the repository (see for instance `sample`). This sample file lists all the options. *Only the keywords listed here will be used, all other words/characters are ignored*. One can check the options used by examining **foo.configuration**, where **foo** is your base output filename.

We suggest the following files as a basis:

- `N-body simulations configuration`
  - This config file is for running a pure N-body simulation, producing 6dfof halos, find substructure and then calculating a variety of properties for each object. The reference position about which quantities are calculated is the minimum potential of an object. Substructure are subhalos, required to be approximately self-bound (particles allowed to have potential energy 0.95 times that of the kinetic energy). There are also similar config files that use `3dfof halos`, one setup to also find `unbound tidal debris`.
- `Hydro simulations configuration`
  - This config is setup to load in all particles from a hydro sim and calculate a variety of quantities. It is similar to the N-body sample.
- `SWIFT N-body simulation configuration`
  - This config is setup to load in a swift snapshot. It is similar to the N-body sample but here is using 3dfof halos.
- `SWIFT Hydro simulation configuration`
  - This config is setup to load in a swift hydro snapshot and also load in extra information from the snapshot to calculate extra hydro/star/bh quantitites. Another example of such a config with specific black hole related quantities is also `available`.

Also provided are config files for the `SURFS` and :download:GENESIS<../examples/genesis2019_configuration.cfg> simulations.

Warning: Note that if misspell a keyword it will not be used.

Warning: Since this file is always written **DO NOT** name your input configuration file **foo.configuration**.

There are numerous key words that can be passed. Here we list them, grouped into several categories:

I/O

Input and output related options

..topic:: Input related

`Cosmological_input = 1/0`

- Flag indicating that input simulation is cosmological or not. With cosmological input, a variety of length/velocity scales are set to determine such things as the virial overdensity, linking length.

`Input_chunk_size = 100000`

- Amount of information to read from input file in one go (100000).

`HDF_name_convention =`

- Integer describing HDF dataset naming convection. Currently implemented values can be found in *HDF Input*.

`Input_includes_dm_particle = 1/0`

- Flag indicating whether file contains dark matter/N-body particles in input file.

`Input_includes_gas_particle = 1/0`

- Flag indicating whether file contains gas particles in input file.

`Input_includes_star_particle = 1/0`

- Flag indicating whether file contains star particles in input file.

`Input_includes_bh_particle = 1/0`

- Flag indicating whether file contains black hole particles in input file.

`Input_includes_wind_particle = 1/0`

- Flag indicating whether file contains wind particles in input file.

`Input_includes_tracer_particle = 1/0`

- Flag indicating whether file contains tracer particles in input file.

`Input_includes_extradm_particle = 1/0`

- Flag indicating whether file contains extra (low resolution) N-body particles in input file from a zoom simulation.

Gas related input

`Gas_internal_property_names = ,`

- Comma separated list of strings listing extra gas properties to be read from HDF file for which bulk mean/total properties are calculated for objects. Useful way of passing properties like molecular H2 fraction, etc.

`Gas_chemistry_names = ,`

- Comma separated list of strings listing extra chemical properties to be read from HDF file for which bulk mean/total properties are calculated for objects. Useful way of passing properties like molecular H2 fraction, etc.

`Gas_chemistry_production_names = ,`

- Comma separated list of strings listing extra production channels for metals to be read from HDF file for which bulk mean/total properties are calculated for objects. Useful way of passing properties like molecular H2 fraction, etc.

Star related input

`Star_internal_property_names = ,`

- Comma separated list of strings listing extra star properties to be read from HDF file for which bulk mean/total properties are calculated for objects. Useful way of passing properties like molecular H2 fraction, etc.

`Star_chemistry_names = ,`

- Comma separated list of strings listing extra chemical properties to be read from HDF file for which bulk mean/total properties are calculated for objects. Useful way of passing properties like molecular H2 fraction, etc.

`Star_chemistry_production_names = ,`

- Comma separated list of strings listing extra production channels for metals to be read from HDF file for which bulk mean/total properties are calculated for objects. Useful way of passing properties like molecular H2 fraction, etc.

Black hole related input

`BH_internal_property_names = ,`

- Comma separated list of strings listing extra black properties to be read from HDF file for which bulk mean/total properties are calculated for objects. Useful way of passing properties like molecular H2 fraction, etc.

`BH_chemistry_names = ,`

- Comma separated list of strings listing extra chemical properties to be read from HDF file for which bulk mean/total properties are calculated for objects. Useful way of passing properties like molecular H2 fraction, etc.

`BH_chemistry_production_names = ,`

- Comma separated list of strings listing extra production channels for metals to be read from HDF file for which bulk mean/total properties are calculated for objects. Useful way of passing properties like molecular H2 fraction, etc.

Extra DM related input

`Extra_dm_internal_property_names = ,`

- Comma separated list of strings listing extra dm properties to be read from HDF file for which bulk mean/total properties are calculated for objects. Useful for modified dark matter simulations, such as annihilating and self-interactive dark matter.

Gadget related input

`NSPH_extra_blocks =`

- Integer inticading the number of extra **SPH** blocks are read in the file if gadget input.

`NStar_extra_blocks =`

- Integer inticading the number of extra **star** blocks are read in the file if gadget input.

`NBH_extra_blocks =`

- Integer inticading the number of extra **BH** blocks are read in the file if gadget input.

Output related

`Output = filename`

- Output base name. Overrides the name passed with the command line argument **-o**. Only implemented for completeness.

`Output_den = filename`

- A filename for storing the intermediate step of calculating local densities. This is particularly useful if the code is not compiled with **STRUCDEN** & **HALOONLYDEN** (see *Compilation Options*).

`Separate_output_files = 1/0`

- Flag indicating whether separate files are written for field and subhalo groups.

`Write_group_array_file = 1/0`

- Flag indicating whether to producing a file which lists for every particle the group they belong to. Can be used with **tipsy** format or to tag every particle.

`Binary_output = 2/1/0`

- **Integer flag indicating type of output.**
  - **2** self-describing binar format of HDF5. **Recommended**.

- **1** raw binary.
- **0** ASCII.

`Extended_output = 1/0`

- Flag indicating whether produce extended output for quick particle extraction from input catalog of particles in structures

`Spherical_overdensity_halo_particle_list_output = 1/0`

- Flag indicating whether particle IDs identified within the spherical overdensity of field halos is written (to a .catalog_SOlist). Useful if looking at evolution of particles within spherical overdensities.

`Sort_by_binding_energy = 1/0`

- Flag indicating whether particle IDs written in .catalog_particles are sorted by binding energy (1) or potential energy (0).

`No_particle_ID_list_output = 1/0`

- Flag indicating whether particle IDs written (i.e., write the .catalog_* files). Default is 1. Particle ID files are necessary for constructing merger trees but if just properties of (sub)halos, then turn off.

Searching for Structures

Options related to searching for (sub)halos. General search parameters set particles to be search and the overall type of search.

`Particle_search_type = 1/2/3/4`

- **An integer describing what types of particles are searched. A full list of options is in *Particle search types*. Typica**
  - **1** *All* particles are searched
  - **2** *DarkMatter* particles (which are typically defined as type 1,2,3 for gadget) are searched
  - **3** Star particles (which are typically defined as type 4 for gadget) are searched
  - **4** Gas particles (which are typically defined as type 0 for gadget) are searched

`Baryon_searchflag = 0/1/2`

- **An integer indicating gas/stellar search done separately from DM search.**
  - **2** field search also altered to treat baryons differently, allowing only DM particles to be used as head links (ie link dm-dm, dm-baryon, but not baryon-baryon nor baryon-dm). Then DM substructure search with baryons associated to closest DM particle in phase-space. **Recommended**.
  - **1** field search run as normal and then substructure search for baryons run using baryons identified in field search.
  - **0** do nothing special for baryon particles.

`Search_for_substructure = 1/0`

- Flag indicating whether field objects are searched for internal substructures. Default is 1 (on)

`Singlehalo_search_search = 0/1`

- Flag indicates that no field search is going to be run and the entire volume will be treated as a background region (halo). Useful if searching for substructures in non-cosmological simulations. But can also be co-opted for other searches using different outlier criteria and FOF algorithms

Parameters related to field (halo) search

`FoF_Field_search_type = 5/4/3`

- **An integer indicating what type of field search is run. There are several**
  - **5** standard 3D FOF based algorithm
  - **4** standard 3D FOF based algorithm **FOLLOWED** by 6D FOF search using the velocity scale defined by the largest halo on particles in 3DFOF groups
  - **3** standard 3D FOF based algorithm **FOLLOWED** by 6D FOF search using *adaptive* velocity scale for each 3DFOF group on particles in these groups.

`Halo_3D_linking_length = 0.2`

- Linking length used to find configuration space 3D FOF halos. If cosmological file then assumed to be in units of inter particle spacing, if loading in a single halo then can be based on average interparticle spacing calculated, otherwise in input units. Default is 0.2 in interpaticle spacing units.

`Halo_velocity_linking_length_factor = 1.0`

- Multiplicative factor of order unity for the dispersions used in 6D searches. Typical values are order unity as velocity dispersions are used to define the velocity linking length scale.

`Halo_6D_linking_length_factor = 1.0`

- Multiplicative factor of order unity that allows one to use different configuration space linking lengths between 3DFOF and 6DFOF field search. Typically this is 1.0

`Halo_6D_vel_linking_length_factor = 1.25`

- Multiplicative factor of order unity scaling applied to dispersions used in 6DFOF field search. Typical values are 1.25.

`Keep_FOF = 0/1`

- Flag that keeps the 3DFOF if field 6DFOF search is done. This is typically invoked when searching for galaxies as the 3DFOF can be interpreted as the inter halo stellar mass and 6DFOF galaxies.

`Minimum_halo_size =-1`

- Integer that allows field objects (or so-called halos) to require a different minimum size than all other substructures. Ignored if not passed or <0, with halos defaulting to `Minimum_size` value.

Parameters related to substructure search

Note: default values are fine and typically do not need to be set in the configuration file. Exception would be Minimum_size

`FoF_search_type = 1`

- Integer indicating what type of FOF algorithm to use. Several substructure FOF criteria are implemented (see *FOF search types* for complete list). Suggested value is **1**, the standard phase-space based, well tested VELOCIraptor criterion.

Outlier_threshold = 2.5

- Threshold of sigma level of outliers to be searched which should be order unity but > 1 (default is 2.5)

Significance_level = 1.0

- Minimum significance level of a substructure which should be order unity (default is 1)

Velocity_ratio = 2.0

- Speed ratio used in linking particles which should be order unity and > 1 (default is 2)

Velocity_opening_angle = 0.10

- Angle between velocities when linking (in units of π) (default is 0.10)

Substructure_physical_linking_length = 0.1

- Physical linking length used in phase-space substructure FOF. If cosmological file then assumed to be in units of inter particle spacing, if loading in a single halo then can be based on average interparticle spacing calculated, otherwise in input units. Default is 0.1 in interpaticle spacing units.

CMrefadjustsubsearch_flag = 1/0

- Flag indicating whether particles are moved to the rough CM velocity frame of the background before substructures are searched for (default is on)

Iterative_searchflag = 1/0

- Flag to use interactive substructure search which is designed to first identify spatially compact candidate outlier regions and then relaxes the criteria to find the more diffuse (in phase-space) regions associate with these candidate structures (default is on)

Iterative_linking_length_factor = 2.0

- Factor multiplied with linking length when using iterative method and identifying outlier regions associated with the initial candidate list of spatially compact outlier groups. Typical values are order `Halo_linking_length_factor` (2.0)

Iterative_threshold_factor = 1.0

- Factor multiplied with threshold when using iterative method and identifying outlier regions associated with the initial candidate list of spatially compact outlier groups. Typical values are order unity.

Iterative_Vratio_length_factor = 1.0

- Factor multiplied with speed ratio when using iterative method and identifying outlier regions associated with the initial candidate list of spatially compact outlier groups. Typical values are order unity.

Iterative_ThetaOp_length_factor = 1.0

- Factor multiplied with opening angle when using iterative method and identifying outlier regions associated with the initial candidate list of spatially compact outlier groups. Typical values are order unity.

Minimum_size = 20

- Minimum number of particles in a (sub)structure (default is 20).

Configuration for local density calculation used to identify substructures

Note: default values are fine and typically do not need to be set in the configuration file.

Local_velocity_density_approximate_calculation = 2/1/0

- **Flag indicating how to calculate computationally expensive local velocity densities.**
  - **2** approximative search limited to particles in halos (requires no mpi communication). **Recommended**.
  - **1** approximative search, group particles in leaf nodes of tree
  - **0** full search per particle.

`Nsearch_velocity = 32`

- Number of velocity neighbours used to calculate velocity density (suggested value is 32)

`Nsearch_physical = 32`

- Number of physical neighbours searched to calculate velocity density (suggested value is 256)

`Cell_fraction = 0.1`

- Fraction of a halo contained in a subvolume used to characterize the background (suggested value is 0.01)

`Grid_type = 1`

- **Integer describing type of grid used to decompose volume for substructure search (suggested value is 1)**
  - **1** standard physical shannon entropy, balanced KD tree volume decomposition into cells. **Recommended**
  - **2** phase phase-space shannon entropy, balanced KD tree volume decomposition into cells
  - **3** simple simple physical balanced KD tree decomposition of volume into cells

Configuration for core search and growth.

This either identifies major mergers in DM simulations or used to find galaxies when searching for stars.

`Halo_core_search = 0/1/2`

- **Integer allows one to explicitly search for large 6D FOF cores that are indicative of a recent major merger. Since sub**
  - **2** search for cores and growth them. **Recommended**.
  - **1**
  - **0** do not search cores.

`Use_adaptive_core_search = 0/1`

- Flag allows one to run complex adaptive phase-space search for large 6D FOF cores and then use these linking lengths to separate mergers. 0 is simple high density dispersively cold cores with velocity scale adaptive, 1 is adaptive in both configuration & velocity.

`Use_phase_tensor_core_growth = 0/1`

- Flag allows one to run complex phase-space growth of merger remnants (6D FOF cores found). 0 is assignment with simple x and v dispersion to nearest core particle, 1 is phase-space tensor distance assignemnt to CM of core.

`Halo_core_ellx_fac =`

- Factor applied to linking length when identifying merger remnants. Typically values are 0.5

`Halo_core_ellv_fac =`

- Factor applied to local dispersion to define the velocity scale used to identify merger remnants. Typically values are order unity

`Halo_core_ncellfac = 0.005`

- Factor used to determine the minimum number of particles a merger remnants is composed of using number of particles in the halo times this factor. For DM typically values are 0.005.

`Halo_core_adaptive_sigma_fac = 2.0`

- Factor used when running fully adaptive core search, specifies the width of the physical linking length in configuration space dispersion (think of this as how many sigma to include). Typically values are 2. This has been tested on hydrodynamnical simulations to separate galaxy mergers.

`Halo_core_num_loops = 10`

- Allows the core search to iterate, shrinking linking lengths used till the number of cores identified reaches zero or this limit is reached. Allows apative search with larger linking length to be robust. Typically values are 10, though typically loops run twice.

`Halo_core_loop_ellx_fac = 0.75`

- Factor by which configuration linking length is decreased when running loops for core search. Typically values are 0.75

`Halo_core_loop_ellv_fac = 1.0`

- Factor by which velocity linking length is decreased when running loops for core search. Typically values are 1.

`Halo_core_loop_elln_fac = 1.2`

- Factor by which min group size is changed when running loops for core search. Typically values are order unity & > 1.

`Halo_core_phase_significance = 2.0`

- Significance a core must be in terms of phase-space distance scaled by dispersions (sigma). Typical values are order unity & > 1.

Configuration for cleaning up substructuers that overlap in phase-space.

Substructures can be merged together if they overlap in phase space.

`Structure_phase_merge_dist = 0.25`

- Phase-distance normalised by dispersions below which structures are merged together. Typical valuse are < 1.

`Apply_phase_merge_to_host = 1`

- Flag whether to also check substructures can be merged with the host background. 1 is on.

Unbinding

Particles in strutures can be checked to see if they are bound relative to a kinetic reference frame (CM of the structure). This cleans the (sub)structures of spurious objects and particles.

Unbind_flag = 1/0

- Flag indciating whether substructures passed through an unbinding routine.

Unbinding_type = 1/0

- Integer setting the unbinding criteria used. Either just remove particles deemeed "unbound" (**1**), that is those with $\alpha T + W > 0$ given by `Allowed_kinetic_potential_ratio`, or (**0**) additionally removes "unbound" and least bound particles till system also has a true bound fraction > `Min_bound_mass_frac`.

Allowed_kinetic_potential_ratio =

- Ratio of kinetic to potential energy at which a particle is still considered bound, ie: particle is still bound if $\alpha T + W < 0$, so $\alpha = 1$ would be standard unbinding and $\alpha < 1$ allows one to identify unbound tidal debris. Given that **VELOCIraptor** was designed to identify tidal streams, it makes little sense to have this set to 1 unless explicitly required. Note that the code still separates particles into bound and unbound. Values of $\alpha \geq 0.2$ seems to minimize the number of false positives in tidal debris while still identifying completely unbound tidal debris.

Min_bound_mass_frac =

- Minimum fraction of particles that must be self-bound. If interested in identifying tidal debris, ues values of 0.2, for self-bound substructures, use $\gtrsim 0.5$

Bound_halos = 0/1/2

- Integer that ignores the boundness of field structures (haloes) (**0**), checks if they are self bound only before (**1**) or also after (**2**) substructures have been identified and extracted from the halo. Demanding boundness after substructure search can have interesting consequences as it is possible that a multiple merger will appear as a single FOF halo, however all with all the cores removed, the FOF halo is actually an unbound structure.

Keep_background_potential = 1/0

- Flag indicating whether while checking if a structure is bound, to treat the candidate structure in isolation, updating the potential continuously, or leave the background potential. background sea. When finding tidal debris, it is useful to keep the background. ref Options.uinfo & ref UnbindInfo.bgpot n

Kinetic_reference_frame_type = 0/1

- Integer that sets the kinetic frame when determining whether particle is bound. Default is to use the centre-of-mass velocity frame (0) but can also use region around minimum of the potential (1).

Min_npot_ref = 10

- The minimum number of particles used to calculate the velocity of the minimum of the potential (default is 10).

Frac_pot_ref = 0.1

- Fraction of particles used to calculate the velocity of the minimum of the potential (0.1). If smaller than `Min_npot_ref`, that is used.

Unbinding_max_unbound_removal_fraction_per_iteration = 0.5

- Maximum fraction of unbound particles removed per iteration in unbinding process.

`Unbinding_max_unbound_fraction = 0.95`

- Maximum fraction of particles that can be considered unbound before group removed entirely and is not processed iteratively.

`Unbinding_max_unbound_fraction_allowed = 0.005`

- Maximum fraction of unbound particles allowed after unbinding. If set to zero, all unbound particles removed.

`Approximate_potential_calculation = 1/0`

- Calculate potentials using significantly faster approximate method (which with standard settings has an erorr 1e-3). Default is 0 (off).

`Approximate_potential_calculation_particle_number_fraction = 0.1`

- Use 0.1 of all particles in object to calculate gravitational potential (values of <0.01 can lead to larger errors, values of >0.2 cause calculation to not be significantly faster than standard calculation).

`Approximate_potential_calculation_min_particle = 5000`

- Use a minimum of 5000 particles in approximate method. Approximate method should only be used for well resolved objects as error increases with less well resolved objects and the speed up is not as significant.

Properties

Configuration options related to the bulk properties calculated.

`Inclusive_halo_mass = 3/2/1/0`

- **Flag indicating whether inclusive masses are calculated for field objects.**
  - **3** indicates inclusive SO masses are calculated after substructure is found.
  - **2** indicates inclusive SO masses are calculated before substructure is found.
  - **1** indicates inclusive SO masses are calculated before substructure is found but limited to particles in the halo.
  - **0** indicates masses exclusive.

`Iterate_cm_flag = 0`

- Flag indicating whether to iteratively find the centre-of-mass of an object (1) or simply determine bulk centre of mass and centre of mass velocity (0). Calculation is based on all particles exclusively belonging to the object.

`Reference_frame_for_properties = 2`

- **Flag indicating what reference position to use when calculating radially dependent properties.**
  - **2** use the position of the particle with the minimum potential.
  - **1** use the position of the most bound particle.
  - **0** use the centre-of-mass.

`Particle_type_for_reference_frames = -1`

- **Flag indicating what particle type is used to determine the minimum potential reference position.**

  - **-1** all particle types
  - **0-6** other int correspond to a specific particle type. For instance 1 would be dark matter particles

`Extensive_halo_properties_output = 1`

- Flag indicating that one should calculate more properties for objects, such as angular momentum in spherical overdensity apertures.

`Extensive_gas_properties_output = 1`

- Flag indicating that in addition to calculating extra halo properties also calculate gas content in spherical overdensity apertures as well as their angular momentum. Must be used in conjunction with `Extensive_halo_properties_output = 1`.

`Extensive_star_properties_output = 1`

- Flag indicating that in addition to calculating extra halo properties also calculate stellar content in spherical overdensity apertures as well as their angular momentum. Must be used in conjunction with `Extensive_halo_properties_output = 1`.

Aperture related config options

`Calculate_aperture_quantities = 1`

- Flag on whether to calculate aperture related masses, dispersions, metallicities

`Number_of_apertures = 6`

- Number of spherical apertures

`Aperture_values_in_kpc = 3,5,10,30,50,100,`

- Comma separated list of values in kpc

`Number_of_projected_apertures = 3`

- Number of projected apertures. Code calculates 3 projections per aperture: x, y, z.

`Projected_aperture_values_in_kpc=10,50,100,`

- Comma separated list of values in kpc

Spherical overdensity related config options

`Number_of_overdensities = 5`

- Number of spherical overdensities

`Overdensity_values_in_critical_density=25,100,500,1000,2500,`

- Comma separated list of spherical overdensity thresholds in units of the critical density in cosmological simulations

Radial profile related config options

`Calculate_radial_profiles = 1`

- Flag on whether to calculate radial profiles of masses

`Radial_profile_norm = 0`

- Flag setting the radial normalisation and scaling. Default is log rad bins, in proper kpc

`Number_of_radial_profile_bin_edges = 9`

- Number of bin edges listed. Assumes lowest bin edge is r=0.

`Radial_profile_bin_edges = -2.,-1.50,-1.00,-0.50,0.00,0.50,1.00,1.50,2.00`

- Comma separated list of (log) r bin edges. Here example is for log r in proper kpc binning so values are log(r).

Configuration for Extra Properties

These are configuration options related to the bulk properties calculated based on extra properties of the particles. For instance, if hydro particles have a field called `Turbulence` that contains some quantity of the internal turbulent energy and one wanted to calculate the average of this value for an object, one would use these options to load data from an HDF5 file (other inputs are not so easily parsed, making this not an option). One needs to provide what calculation to do (in the form of an integer flag specifying the calculation) and a string indicating the units. If the input is in the form of a 2D array from which a particular column is to be used, one can also set an index. The result is sorted in an output field that contains the name of the input field, the index (if >0), and a simple string describing the function and ending with particle type, ie: `Turbulence_average_gas` These config options are combinations of particle type, categories and entry types. A full entry must be provided in a comma separated list and terminate in a comma.

- **Currently implemented are options for**
  - `Gas_`
  - `Stars_`
  - `BH_`
  - `Extra_DM_`
- **The currently catagories of properties are (except for `Extra_DM` which only has the first listed). An inpu field can be spe**
  - `_internal_property`
  - `_chemistry`
  - `_chemistry_production`
- **The entries are**
  - `_names`
  - `_index_in_file`
  - `_calculation_type`
  - `_input_output_unit_conversion_factors`
  - `_output_units`

Calculations allowed are as follows. You can add **massweighted** to any entry to calculate the mass weighted quantity. Note at entries should be lower case.

- average
- total
- std (standard deviation)
- min
- max
- logaverage (average(log(x)))
- logstd (std(log(x)))

Output units are indices of standard units separated by colons along with any additional extra units which are added as strings to the name of the output. The standard units for which indices can be provided are

- Mass (where conversion to solar mass provided can be used to convert output to known units)
- Length (where conversion to kpc provided can be used to convert output to known units)
- Velocity (where conversion to km/s provided can be used to convert output to known units)
- Time (where conversion to Gyrs provided can be used to convert output to known units)

Thus to specify mass per unit time^2 and another entry with force, as an example, one would use a string of

- "1:0:0:-2:,1:0:1:-1:,"

This does require the input to be converted appropriately to match the units of mass, length, velocity, time. This attribute information will be stored the attributes associated with the data set, similar to other fields. One can also provide complex units with a string that will be stored in a attribute **Dimension_Extra_Info**

- "cookies_per_person,"

One can also calculate total or average in apertures provided aperture quantities are being calculated.

- aperture_total
- aperture_average

Example extra hydro Properties related config options

`Gas_internal_property_names = ,`

- Names of fields to be read from an input HDF5 file that relate to hydro quantities, for which calculations can be done

`Gas_internal_property_index_in_file = ,`

- Index in 2d array to be read from an input HDF5, useful for fields like metallicity where it is common to have an entry for each element

`Gas_internal_property_calculation_type = ,`

- Integer flag indicating what calculation is to be done.

`Gas_internal_property_input_output_unit_conversion_factors = ,`

- Float storing the conversion factor (if not 1.0) to take input units to output units.

`Gas_internal_property_output_units = ,`

- String storing the units of the output.

Simulation Info

Options related to the input and output units and cosmology.

Units

Set internal (and output) units and conversion factors to well known units

`Length_input_unit_conversion_to_output_unit =`

- Factor by which input length unit is scaled, setting the internal code and output unit

Velocity_input_unit_conversion_to_output_unit =

- Factor by which input velocity unit is scaled, setting the internal code and output unit

Mass_input_unit_conversion_to_output_unit =

- Factor by which input mass unit is scaled, setting the internal code and output unit

Metallicity_input_unit_conversion_to_output_unit =

- Factor by which input metallicity unit is scaled, setting the internal code and output unit

Star_formation_rate_input_unit_conversion_to_output_unit =

- Factor by which input star formation rate of gas unit is scaled, setting the internal code and output unit

Stellar_age_input_unit_conversion_to_output_unit =

- Factor by which input stellar ages unit is scaled, setting the internal code and output unit

Stellar_age_input_is_cosmological_scalefactor =

- 0/1 to indicate whether stellar age is cosmological scale factor

Star_formation_rate_input_is_specific_star_formation_rate =

- 0/1 to indicate whether star formation rates are specific star formation rates

Gas_star_forming_rate_threshold =

- Value in code units that splits gas from star forming and non-star forming. Default value is 0

Gravity =

- Gravity in the internal output units, that is should be set such that $v^2 = Gm/r$, where v,m,r are the internal velocity, mass and length units. Note that this does not have to be provided as it will be calculated based on the output units (that indicate how they are converted to kpc, km/s etc) and the gravitational constant of 6.67430e-11 kg^-1 m^3 / s^2. A warning will be given if the provided gravitational constant differs significantly from the expected value given the output.

Hubble_unit =

- Unit of Hubble expansion in internal output units (from normal km/s/Mpc use 100). Like the gravitational constant, this does not have to be provided as it will be calculated from the output units. A warning will be given if provided value differs significantly from the expected value given the output. This is ignored if non-cosmological input

Mass_value =

- If code is compiled not to store mass using the option **NOMASS** (see *Compilation Options*) then set this value.

Length_unit_to_kpc =

- Specify the conversion factor from the output unit to kpc

Velocity_unit_to_kms =

- Specify the conversion factor from the output unit to km/s

Mass_unit_to_solarmass =

- Specify the conversion factor from the output unit to solar masses

Stellar_age_to_yr =

- Specify the conversion factor from the output unit to yr

`Comoving_units = 1/0`

- Flag indicating whether the properties output is in physical or comoving little h units.

Cosmology

If input is cosmological, then for some input formats (gadget, HDF), these quantites can be read from the input file. Tipsy formats require that these be set in the configuration file.

`Period = 0`

- Period of the box in input units.

`Scale_factor = 1.0`

- Scale factor time

`h_val = 1.0`

- The "little h" value often used in cosmological simulations.

`Omega_m = 1.0`

- Matter density in units of the critical density at z=0 used in cosmological simulations.

`Omega_Lambda = 0.0`

- Energy density of the cosmological constant (or dark energy) in units of the critical density at z=0 used in cosmological simulations.

`Omega_cdm = 1.0`

- Dark matter density in units of the critical density at z=0 used in cosmological simulations. For non-standard DM models (annihilating, decaying, coupled), may be useful to provide the current DM density.

`Omega_b = 0.0`

- Baryon density in units of the critical density at z=0 used in cosmological simulations.

`Omega_r = 0.0`

- Radiation density in units of the critical density at z=0 used in cosmological simulations. Typically 0 (negligible).

`Omega_nu = 0.0`

- Neutrino density in units of the critical density at z=0 used in cosmological simulations. Typically 0 (negligible).

`Omega_k = 0.0`

- Curvature density in units of the critical density at z=0 used in cosmological simulations. Typically 0 (flat).

`Omega_DE = 0.0`

- Dark Energy density in units of the critical density at z=0 used in cosmological simulations. This is addition to (or replacing) the energy density of the cosmological constant and has an associated equation of state, w_{DE}.

`w_of_DE = -1.0`

- Equation of state of the dark energy fluid, $w = \frac{p}{\rho}$. This is not necessary unless one is using a cosmological simulation with $w \neq -1$. Currently not fully implemented.

`Virial_density = 200.0`

- Virial overdensity in units of the background matter density used in cosmological simulations. If -1, then the Bryan & Norman 1998 virial density is calculated based on a LCDM cosmology, otherwise overrides the Bryan & Norman calculation.

`Critical_density = 1.0`

- Critical density in input units used in cosmological simulations.

Parallel

Options related to MPI/OpenMP/Pthread parallelisation.

MPI

MPI specific options

`MPI_part_allocation_fac = 0.1`

- Factor used in memory allocated in mpi mode to store particles is (1+factor)* the memory need for the initial mpi decomposition. This factor should be >0 and is mean to allow a little room for particles to be exchanged between mpi threads withouth having to require new memory allocations and copying of data.

`MPI_particle_total_buf_size =`

- Total memory size in bytes used to store particles in temporary buffer such that particles are sent to non-reading mpi processes in chunks of size buffer_size/NProcs/sizeof(Particle).

`MPI_number_of_tasks_per_write =`

- Number of mpi tasks that are grouped for collective HDF5 writes is parallel HDF5 is enabled. Net result is that the total number of files written is ceiling(Number of MPI tasks)/(Number of tasks per write)

`MPI_use_zcurve_mesh_decomposition = 1/0`

- Whether to use a z-curve spatial decomposition (advised). Default is true

`MPI_zcurve_mesh_decomposition_min_num_cells_per_dim =`

- Minimum number of cells per dimension from which to construct a mesh used in the z-curve decomposition. Min number is 8. Code does use

number of processors to scale mesh resolution using NProcs^(1/3)*2 if > 8. For zooms, advised to set this to a high value corresponding to the order of a few times Lbox/Zoom_region_length.

OpenMP specific parallelisation options

`OMP_run_fof = 1`

- Flag indicating whether to run FOF searches with OpenMP threads.

`OMP_fof_region_size = 100000000`

- Number of particles per OpenMP region.

Miscellaneous

Other configuration options

Snapshot_value =

- If halo ids need to be offset to some starting value based on the snapshot of the output (say to make temporally unique halo ids that are useful for some halo merger tree codes), one can specific a snapshot number. All halo ids will be listed as internal haloid + snapnum * 10^{12} (or if using 32 bit integers and 64 bit integers, then ids offset by 10^6).

Effective_Resolution =

- If running a multiple resolution zoom simulation, simple method of scaling the linking length by using the period and this effective resolution, ie: $p/N_{\rm eff}$

Verbose = 0/1/2

- Integer indicating how talkative the code is (2 very verbose, 1 verbose, 0 quiet).

Particle search types

List of particle types (and combinations) that can be searched are

group **SEARCHTYPES**

Defines

PSTALL

PSTBH

PSTDARK

PSTGAS

PSTNOBH

PSTSTAR

FOF search types

List of fof aglorithms implemented are

group **FOFTYPES**

Defines

FOF3D
3d search

FOF6D
6d fof but only use single velocity dispersion from largest 3d fof object

FOF6DADAPTIVE
no subsets made, just 6d (with each 6dfof search using 3d fof velocity dispersion,)

FOF6DCORE

FOF6DSUBSET
6D FOF search but only with outliers

FOFBARYON6D
baryon 6D FOF search

FOFBARYONPHASETENSOR
baryon phase tensor search

FOFSTNOSUBSET
phase-space FOF but no subset produced

FOFSTPROB
call FOFStreamwithprob

FOFSTPROBLX
like FOFStreamwithprob search but here linking length adjusted by velocity offset, smaller lengths for larger velocity offsets

FOFSTPROBNN
like FOFStreamwithprob search but search is limited to nearest physical neighbours

FOFSTPROBNNLX
like *FOFSTPROBLX* but for NN search

FOFSTPROBNNNODIST
like *FOFSTPROBNN* but there is not linking length applied just use nearest neighbours

FOFSTPROBSCALEELL

FOFSTPROBSCALEELLNN

HDF Input

List of naming conventions are

group **HDFNAMES**

Defines

HDFEAGLENAMES

HDFEAGLEVERSION2NAMES

HDFGADGETXNAMES

HDFGIZMONAMES

HDFILLUSTISNAMES

HDFMUFASANAMES

HDFNUMNAMETYPES

HDFOLDSWIFTEAGLENAMES

HDFSIMBANAMES

HDFSWIFTEAGLENAMES

HDFSWIFTFLAMINGONAMES

For complete discussion of implementation see `../src/hdfitems.h`

CHAPTER

THREE

UNDERSTANDING AND ANALYSING VELOCIRAPTOR OUTPUT

VELOCIraptor produces several different types of output files.

(with the mpi threads appending their rank to the end of the file name unless not compiled with MPI or if Parallel HDF5 is used.):

Standard files

- `.properties`: a file containing the bulk properties of all structures identified.
- `.catalog_groups`: a file containing the size of the structures (in number of particles associated) & information need to read particle information produced by velociraptor
- `.catalog_particles`: a file containing a list of particle IDs of those in structures. Information contained in `.catalog_groups` is used to parse this data.
- `.catalog_particles.unbound`: similar to `catalog_particles` but lists particles in structures but are formally unbound. Information contained in `.catalog_groups` is used to parse this data.

Extra files

- `.catalog_parttypes`: a file similar to `.catalog_particles` but containing a list of particle types of those in structures. Information contained in `.catalog_groups` is used to parse this data. Produced if multiple particle types are processed by **VELOCIraptor**.
- `.catalog_parttypes.unbound`: similar to `catalog_parttypes` but lists particles in structures but are formally unbound.
- `.profiles` : a file containing the radial profiles of groups. Produced if radial profiles are requested.
- `.catalog_SOlist` : a file containing the a list of particle IDs of particles found within a large Spherical region around Field halos. Produced if a list of paritcles wihtin so regions is requested.

3.1 Properties

There are a variety of properties calculated for each object found. Some are typical of all halo finders such as the mass of an object (which can be a halo, subhalo, tidal debris), along with more complex properties such as the eigenvectors and eigenvalues of the mass distribution defined by the reduced inertia tensor. The number of properties also varies with the type of run. For hydrodynamic simulations where **VELOCIraptor** has been compiled to use gas, star and black hole properties, such as masses, temperatures, etc are also calculated. The code will also calculate properties based on loading specific extra fields associated with particle types (this interface requires HDF5 input or on the fly invocation and outputs properties with the same name as the loaded property, see *Using VELOCIraptor*).

Note that if HDF5 output is produced, the properties will be in the form of data sets with specific names and each data set will have attributes describing the unit of the field in the form of **Dimension_Length**, **Dimension_Mass**, **Dimension_Velocity**, **Dimension_Time**, which indicate the index of the unit. Extra output from arbitrary input fields can also have unusual units stored in **Dimension_Extra_Info** as a string.

We give an almost complete list of properties and the keyword associate with the property (in ASCII and HDF5). For clarity we list properties in several tables corresponding to

- *Standard Properties*,
- *Gas Properties*,
- *Star Properties*,
- *Black Hole Properties*,
- *Interloper Properties*,
- *Extra DM Properties*,

3.1.1 Standard Properties

This is a list of standard properties that are always calculated unless indicated otherwise (some require an extra configuration option). Properties are calculated relative to the object's centre, which can be either the position of the particle with the minimum potential, or centre-of-mass, or position of most bound particle.

Name	Comments
ID and Type information	
ID	Halo ID. ID = index of halo + 1 + TEMPORALHALOIDVAL * Snapshot_
ID_mbp	Particle ID of the most bound particle in the group.
hostHaloID	ID of the host field halo. If an object is a field halo, this is -1.
Structuretype	Structure types contain information on how the object was found and at wh
numSubStruct	Number of substructures. Subhalos can have subsubhalos.
Mass and radius properties: *All properties are in output units.*	
npart	Number of particles belonging exclusively to the object.
Mass_tot	Total mass of particles belonging exclusively to the object,
	$M_{\rm tot}$.
Mass_FOF	Total mass of particles in the FOF, $M_{\rm FOF}$. Is zero for substructure.
Mass_200mean	Overdensity mass defined by mean matter density, $M_{200\rho_m}$. For field halos,
Mass_200crit	Overdensity mass defined by critical density, $M_{200\rho_c}$. Behaviour like Mass_
Mass_BN98	Overdensity mass defined by mean matter density and $\Delta(z)$ given by Bryan
Mvir	User defined virial mass, $M_{\rm vir}$. Behaviour like Mass_200mean.
R_size	Maximum distance of particles belonging exclusively to the object and the
R_200mean	Radius related to overdensity mass Mass_200mean.

Name	Comments
R_200crit	
R_BN98	
Rvir	
R_HalfMass	Half mass radius based on the Mass_tot.
R_HalfMass_200mean	Half mass radius based on the Mass_200mean.
R_HalfMass_200crit	
R_HalfMass_BN98	
Angular Momentum in Spherical Overdensity: *Calculate if extra halo properties are requested by setting the config option `**Extensive_halo_properties_output=1** `Also calculates inclusive spherical overdensity and also exclusive to halo as _exclusive.*	
Lx_200c	x component of the total angular momentum all the mass within $R_{200\rho_c}$.
Ly_200c	
Lz_200c	
Lx_200m	x component of the total angular momentum all the mass within $R_{200\rho_m}$.
Ly_200m	
Lz_200m	
Lx_BN98	x component of the total angular momentum all the mass within R_{BN98}.
Ly_BN98	
Lz_BN98	
Position and Velocity: *All properties are in output units. Objects have positions periodically wrapped.*	
Xc	x coordinate of centre-of-mass.
Yc	
Zc	
Xcmbp	x coordinate of most bound particle.
Ycmbp	
Zcmbp	
Xcminpot	x coordinate of the minimum potential.
Ycminpot	
Zcminpot	
VXc	v_x velocity of centre-of-mass.
VYc	
VZc	
VXcmbp	v_x velocity of most bound particle.
VYcmbp	
VZcmbp	
VXcminpot	v_x velocity of the particle with the minimum potential.
VYcminpot	
VZcminpot	
Velocity and Angular Momentum: *All properties are in output units.*	
Vmax	Maximum circular velocity based on particles belonging exclusively to the
Rmax	Radius of maximum circular velocity.
sigV	Velocity dispersion based on the velocity dispersion tensor $\sigma_v = \|\Sigma\|^{1/6}$, wh
veldisp_xx	The x, x component of the velocity dispersion tensor.
veldisp_xy	
veldisp_xz	
veldisp_yx	
veldisp_yy	
veldisp_yz	
veldisp_zx	

Name	Comments
veldisp_zy	
veldisp_zz	
Lx	x component of the total angular momentum about the object's centre and c
Ly	
Lz	
lambda_B	Bullock et al (2001) like spin parameter λ_B using total angular momentum
Krot	Measure of rotational support about the angular momentum axis $\kappa_{\rm rot} = \frac{\sum_i}$
Morphology: *All properties are in output units.*	
cNFW	Calculated assuming an NFW profile (Navarro, Frenk, & White 1997) follo
cNFW_200crit	Calculated assuming an NFW profile (Navarro, Frenk, & White 1997) using
cNFW_200mean	
cNFW_BN98	
q	We calculate the shape using the reduced inertia tensor (Dubinski et al, 199
s	Minor axis ratio.
eig_xx	Eigenvectors of morphology.
eig_xy	
eig_xz	
eig_yx	
eig_yy	
eig_yz	
eig_zx	
eig_zy	
eig_zz	
Energy: *All properties are in output units.*	
Ekin	The total kinetic energy, $\sum T_i$.
Epot	The total gravitational potential energy $1/2 \sum W_i$, where 1/2 comes from d
Efrac	The fraction of particles that are formally bound (i.e., have $W_i + T_i < 0$).
Quantities within $R(V_{\rm max})$: Properties based on particles within $r \leq R(V_{\rm max})$.	
RVmax_sigV	Dispersion, like sigV for $r \leq R(V_{\rm max})$.
RVmax_veldisp_xx	Dispersion tensor, like veldisp_xx for $r \leq R(V_{\rm max})$.
RVmax_veldisp_xy	
RVmax_veldisp_xz	
RVmax_veldisp_yx	
RVmax_veldisp_yy	
RVmax_veldisp_yz	
RVmax_veldisp_zx	
RVmax_veldisp_zy	
RVmax_veldisp_zz	
RVmax_lambda_B	Spin parameter, like lambda_B for $r \leq R(V_{\rm max})$.
RVmax_Lx	Total angular momentum, like Lx for $r \leq R(V_{\rm max})$.
RVmax_Ly	
RVmax_Lz	
RVmax_q	Semi-major axis ratio, like q for $r \leq R(V_{\rm max})$.
RVmax_s	Minor axisratio, like s for $r \leq R(V_{\rm max})$.
RVmax_eig_xx	Eigenvectors of morphology, like eig_xx for $r \leq R(V_{\rm max})$.
RVmax_eig_xy	
RVmax_eig_xz	
RVmax_eig_yx	

Name	Comments
RVmax_eig_yy	
RVmax_eig_yz	
RVmax_eig_zx	
RVmax_eig_zy	
RVmax_eig_zz	
Additional Spherical Overdensity Mass/radius: *If extra spherical overdensity values are requested via* Overdensity_values_in_critical_density *config option, code calculates masses/radii/angular momentum following a naming convention of* SO_property_rhocrivalue_rhocrit *where rhocritvalue is the overdensity value in units of the critical density, e.g.,* SO_mass_100_rhocrit. *The code will also calculate quantities based on particle type: gas, star, interloper, following* SO_property_parttype_rhocrivalue_rhocrit	
mass	Mass enclosing a average density of the associated SO value.
Lx	Angular momentum of enclosed mass in x-direction
Ly	in y-direction
Lz	in z-direction
Aperture quantities: *If aperture quantities are requested code calculates a variety of properties within spherical aperture in pkpc. Naming convention is* Aperture_quantity_radiusvalue_kpc, *or for a specific particle type* Aperture_quantity_parttype_radiusvalue_kpc, *e.g.* Aperture_mass_10_kpc. *Particle types where individual quantities are calculated: gas, star, bh, interloper. We list the property names here.*	
mass	Total mass in aperture.
npart	Total number of particles.
rhalfmass	Radius enclosing half the mass within the aperture.
veldisp	Velocity disperion
Projected aperture quantities: *Similar to aperture quantitites but for 3 different projections based on particles within a projected radius in pkpc. Naming convention is* Projected_aperture_i_quantity_radiusvalue_kpc, *where i is from 0, 1, 2 for a x,y,z projection.*	
mass	Total mass in aperture.
rhalfmass	Radius enclosing half the mass within the aperture.

3.1.2 Gas Properties

This is a list of gas properties that are calculated if code is compiled with **USE_GAS**. Some require an extra configuration option. Also, Spherical overdensity masses + angular momentum, aperture properties, projected aperture properties are calculated for gas particles along along with some extra gas only properties.

Name	Comments
Gas quantities: *Bulk properties of gas particles/tracers when compiled to process gas properties. Properties unique to gas are T_gas and SFR_gas.*	
n_gas	Number of gas particles.
M_gas	Total gas mass $M_{\rm gas}$.
M_gas_Rvmax	Gas mass within $R(V_{\rm max})$.
M_gas_30kpc	Gas mass within 30 pkpc.
M_gas_500c	Gas mass within a spherical overdensity of $500\rho_c$.
Xc_gas	x coordinate of centre-of-mass of gas particles relative to Xc.
Yc_gas	
Zc_gas	
VXc_gas	x coordinate of centre-of-mass velocity of gas particles relative to VXc.
VYc_gas	
VZc_gas	

continues on next page

Table 2 – continued from previous page

Name	Comments
Efrac_gas	Like Efrac but for gas particles only.
R_HalfMass_gas	Like R_HalfMass but for gas particles only.
veldisp_xx_gas	Like veldisp_xx but for gas particles only and relative to the centre-of-mass
veldisp_xy_gas	
veldisp_xz_gas	
veldisp_yx_gas	
veldisp_yy_gas	
veldisp_yz_gas	
veldisp_zx_gas	
veldisp_zy_gas	
veldisp_zz_gas	
Lx_gas	Like Lx but for gas particles only and relative to the centre-of-mass.
Ly_gas	
Lz_gas	
q_gas	Like q but for gas particles only and relative to the centre-of-mass.
s_gas	Like s but for gas particles only and relative to the centre-of-mass.
eig_xx_gas	Like eig_xx but for gas particles only and relative to the centre-of-mass.
eig_xy_gas	
eig_xz_gas	
eig_yx_gas	
eig_yy_gas	
eig_yz_gas	
eig_zx_gas	
eig_zy_gas	
eig_zz_gas	
Krot_gas	Like Krot but for gas particles only and relative to the halo's centre.
T_gas	Average temperature of gas.
Zmet_gas	Average metallicity of gas.
SFR_gas	Total star formation rate of gas.
Star Forming (sf)/Non Star Forming (nsf) Gas quantities: *Similar to gas properties but split by sf/nsf gas. For brevity, we list only quantities unique to sf, as the nsf gas is similar but with _nsf naming convention. Only calculated if* **USE_GAS** *and* **USE_STAR** *flags on.*	
M_gas_sf	Total gas mass $M_{\rm gas}$.
R_HalfMass_gas_sf	Half mass radii.
sigV_gas_sf	Velocity dispersion of the gas.
Lx_gas_sf	Like Lx_gas but for star forming gas.
Ly_gas_sf	
Lz_gas_sf	
Krot_gas_sf	Like Krot_gas but for star forming gas
T_gas_sf	Average temperature of star forming gas.
Zmet_gas_sf	Average metallicity of star forming gas.
Aperture quantities: *If aperture quantities are requested code calculates a variety of properties within spherical aperture in pkpc. Naming convention is* Aperture_quantity_gas_radiusvalue_kpc. *We list the additional properties calculated for gas here (which are in addition to mass,rhalfmass, etc).*	
Zmet	Average gas metallicity in aperture.
SFR	Total star formation rate of gas in aperture.

continues on next page

Table 2 – continued from previous page

Name	Comments
Projected aperture quantities: *Similar to aperture quantitites but for 3 different projections based on particles within a projected radius in pkpc. Naming convention is* Projected_aperture_i_quantity_gas_radiusvalue_kpc, *where i is from 0, 1, 2 for a x,y,z projection. We list the additional properties calculated for gas here (which are in addition to mass,rhalfmass, etc).*	
Zmet	Average gas metallicity in projected aperture.
SFR	Total star formation rate of gas in projected aperture.

Name	Comments
Extra Gas Properties: *If extra gas fields are loaded by listing them using* Gas_internal_property_names Gas_chemistry_names *and/or* Gas_chemistry_production_names. *The are associated input options related to the input index calclation type done and output units. The output will have the following naming convention:* nameoffield_index_#_calculation_units_gas *e.g.* `, AlphaElements_index_0_average_unitless_gas. *Also requires that code is compiled with the* **USE_GAS** *flag As an example we show the fields if* Gas_internal_property_names=Pressure,MetalMassFractionFromSNIa, Gas_internal_property_index=0,1, Gas_internal_property_output_units=kPa,unitless, Gas_internal_property_calculation_type=max,average,	
Pressure_index_0_max_kPa_gas	maximum pressure of gas in object.
MetalMassFractionFromS-NIa_index_1_average_unitless_gas	average of this field.
One can also specify aperture_total *and* aperture_average *as functions if aperture quantities are calcualed. The output will have a simlar naming convention to above but with* **Aperture_** *at the start and ending with the aperture aperture itself* #_kpc` for each aperture listed.	

3.1.3 Star Properties

This is a list of stellar properties that are calculated if code is compiled with **USE_STAR**. Some require an extra configuration option.

Name	Comments
Star quantities: *Bulk stellar properties when compiled to process star properties. Similar to gas properties but has _star instead of _ gas. For brevity, we list only quantities unique to star particles.*	
tage_star | Average stellar age.	
Aperture quantities: *If aperture quantities are requested code calculates a variety of properties within spherical aperture in pkpc. Naming convention is* Aperture_quantity_star_radiusvalue_kpc. *We list the additional properties calculated for star here (which are in addition to mass,rhalfmass, etc).*	
Zmet	Average stellar metallicity in aperture.
Projected aperture quantities: *Similar to aperture quantitites but for 3 different projections based on particles within a projected radius in pkpc. Naming convention is* Projected_aperture_i_quantity_star_radiusvalue_kpc, *where i is from 0, 1, 2 for a x,y,z projection. We list the additional properties calculated for gas here (which are in addition to mass,rhalfmass, etc).*	
Zmet	Average stellar metallicity in projected aperture.
Extra Star Properties: *Like the extra gas properties but calculated if ` Star_internal_property_names Star_chemistry_names `and/or* Star_chemistry_production_names. *Naming convention is the same but ends with _star Also requires that code is compiled with the* **USE_STAR** *flag*	

3.1.4 Black Hole Properties

This is a list of black hole properties that are calculated if code is compiled with **USE_BH**. Some require an extra configuration option.

Name	Comments
Black hole quantities: *Bulk properties of black hole particles when compiled to process black hole properties.*	
n_bh	Number of black hole particles.
Mass_bh	Total mass of black hole particles.
Extra Black hole Properties: *Like the extra gas properties but calculated if `BH_internal_property_names BH_chemistry_names `and/or* BH_chemistry_production_names. *Naming convention is simialr save ends with _bh Also requires that code is compiled with the* **USE_BH** *flag*	

3.1.5 Interloper Properties

This is a list of interloper DM properties that are calculated if code is compiled with **ZOOM_SIM**. These properties are based on low resolution particles and can be used to gauge the level of contamination

Name	Comments
Interloper particles: *If analysing multi-resolution simulations, low resolution particles are often treated as contaminants. These are bulk properties of low resolution contaminant particles.*	
n_interloper	Number of low resolution, interloper particles.
Mass_interloper	Total mass of low resolution, interloper particles.

3.1.6 Extra DM Properties

This is a list of Extra DM properties that are calculated if code is compiled with **USE_EXTRADM**. These properties are useful if running on standard dark matter.

Name	Comments
Extra DM Properties: *Like the extra gas properties but calculated if `Extra_DM_internal_property_names `has fields specified. Useful for nonstandard dark matter runs, such as annihilating or interacting dark matter. Naming convention is nameoffield_extra_dm Also requires that code is compiled with the* **USE_EXTRADM** *flag*	

CHAPTER

FOUR

DEVELOPING VELOCIRAPTOR

VELOCIraptor is an freely available from github. It is in active development, with additions for hydrodynamical inputs, extra inputs and functionality being implemented. The code can also be called as a library for on-the-fly halo finding integration into any code. Currently there are hooks for SWIFTSIM.

4.1 Integration into N-Body/Hydro

INDEX

F

H

P