Creation of initial conditions

Galstep

Creates initial conditions for galaxies.

First version was made by Raffael Ruggiero, but I changed some parameters and tweaked the code in order to write for Gadget-4 HDF5 and work in python3.

My version can be found here and quickly downloaded using the terminal with

```
wget https://github.com/elvismello/galstep/archive/refs/heads/master.tar.gz
```

Clustep

Creates initial conditions for galaxy clusters.

First version was made by Raffael Ruggiero, but I changed some parameters and tweaked the code in order to write for Gadget-4 HDF5 and work in python3.

My version can be found here and quickly downloaded using the terminal with

```
wget https://github.com/elvismello/clustep/archive/refs/heads/master.tar.gz
```

snapshotJoiner

It joins snapshots.

Code can be found here.

```
wget
https://github.com/elvismello/snapshotJoiner/archive/refs/heads/master.tar.
gz
```

Programs for manually reading and writing initial conditions

UNSIO

A suitable substitute for `pygadgetreader` and (possibly) `pynbody` for reading Gadget-2 snapshots. It can also be used to write snapshots, which can be quite useful.

It was made by the same author of Glnemo2, Jean.

```
pip install python-unsio
```

There are versions for Fortran and C/C++.

Scripts

Rubens has written some basic scripts that use UNSIO specifically with Gadget-2, which can be found here.

To automatically get a tarball:

```
wget
https://github.com/regmachado/SnapGadget/archive/refs/heads/main.tar.gz
```

Formats

Can be used to read

- Gadget 1;
- Gadget-2;
- Gadget-3 (hdf5);
- Nemo;
- Ramses;
- Misc (List of files, sqlite3 database);

and write

- Gadget-2;
- Gadget-3 (hdf5);
- Nemo.

User guides

For reading files:

- https://projets.lam.fr/projects/unsio/wiki/PythonReadDataNew

For writing files:

- https://projets.lam.fr/projects/unsio/wiki/PythonWriteDataNew

Quick reference

Tags used by UNSIO to indentify each snapshot component:

Component tag	Description

Component tag	Description
gas	Gas particles
halo	Dark matter particles
dm	Dark matter particles
disk	Old stars particles
stars	Stars particles
bulge	Bulge particles
bndry	bndry particles

Properties present in each component

tag string	descripton	numpy data_array passed as parameter
pos	particles positions	numpy 1D array of particles position (size=n*3)
vel	particles velocities	numpy 1D array of particles velocitie (size=n*3)
mass	particles masses	numpy 1D array of particles velocitie (size=n)
acc	particles accelerations	numpy 1D array of particles acceleration (size=n*3)
pot	particles potential	numpy 1D array of particles potential (size=n)
rho	particles densities	numpy 1D array of particles density (size=n)
hsml	particles hsml	numpy 1D array of particles hydro smooth length (size=n)
temp	particles temperatures	numpy 1D array of particles temperature (size=n)
age	particles ages	numpy 1D array of particles age (size=n)
metal	particles metallicity	numpy 1D array of particles metallicity (size=n)
u	particles internal energy	numpy 1D array of particles internal energy (size=n)
aux	particles auxilliary (NEMO)	numpy 1D auxiliary array (size=n)
keys	particles keys (NEMO)	numpy 1D keys array (size=n)
id	particles indexes (NEMO)	numpy 1D keys array (size=n)

Pynbody

A recent development that is supposed to be "The One Program to Analyze Them All". It doesn't always work as intended for non-cosmological Gadget simulations (because of units) and wrongly recognizes Gadget-4 HDF5 as Arepo snapshots. Can write basic snapshots in some formats.

It is still quite simple to be used and can produce quick results and great visualizations, being useful in a pinch.

It can also run with multiple threads and has many sections written in C/C++, being surprisingly fast even though it uses mainly Python.

```
pip install pynbody
```

User guides

Tutorials:

- https://pynbody.github.io/pynbody/tutorials/tutorials.html

Pynbody framework specifics:

- https://pynbody.github.io/pynbody/reference/essentials.html

H5PY

Python API that interfaces with HDF5 libraries. Can be used for reading and writing in HDF5.

Most of the time, it is as dificult to use as pynbody or unsio.

```
pip install h5py
```

Quick Reference

For Gadget HDF5 snapshots, the structure is almost the same for all versions. Using HDF5 everything is loaded as dictionaries, which make things quite a bit easier.

Usual structure and tags are, for the main datasets:

Component tag	**Description**
PartType0	Gas particles
PartType1	Dark matter/Halo particles
PartType2	Disk particles
PartType3	Bulge particles
PartType4	Stars particles
PartType5	Boundary particles

where all particles have the properties:

Property tag	**Description**
Masses	Masses

Property tag	Description
ParticleIDs	IDs
Coordinates	Cordinates as 3-dimensional vectors
Velocities	Velocities as 3-dimensional vectors

and gas particles (PartType0) also have the properties:

Property tag	Description
SmoothingLength	Smoothing length of the particle
Density	Density
StarFormationRate	Star formation rate, if star formation was activated
ElectronAbundance	Electron abundance, if cooling was activated
Metallicity	Gas metallicity, if supernovae feedback was activated
InternalEnergy	Gas internal energy

Simulation programs

Gadget 2

The current most used (for our group) n-body simulations program. It's quite robust and has a lot of features, being useful for numerous needs.

Gadget 4

A newer Gadget version. It has several improvements over Gadget 2 and 3, including some optimizations, better accuracy and scallability.

It natively includes simpler, but ok, methods of star formation and cooling.

Has a cap of 2 GB per file in the Gadget 2 format. The manual advises to use HDF5, since the binary file is old and a bit outdated.

Others

Gadget 3

An updated version of Gadget 2 with numerous improvements. Should function in a similar way to Gadget 4, but, as Volker Springel put it, this code is confuse by times and not very expandable. In Gadget 4 the code should be clearer and easier to modify.

Ramses

"Grid base hydrodynamical solver with adaptive mesh refinement". Has advantages when dealing with hydrodynamics, but is more complex to use.

Similarly to Arepo, this code discretizes the space, basically leaving a given maximum of mass for an element of space, differing from Gadget, where the mass is discretized in the form of particles.

Arepo

It uses a moving mesh, optimizing space to subdivide the mass. It was developed to be an efficient code structure with state of the art numerical techniques, coded in a way that any conmputational astrophysicist could develop their own specialized modules.

Warnings

- Sometimes Gadget-4 snapshots written in the binary format will have strange fields that not every program will be able to read.
- Even though Gadget-4 understands the binary format from Gadget-2, it will not accept it as initial conditions, unless a block of metallicity is inserted into the format for gas particles (I think). Python 3 versions of galstep and clustep will usually work.
    - UNSIO has a way to use this block, reading and writing to it, but it didn't always work the last time I checked.
- HDF5 snapshots from Gadget-2/3 and Gadget-4 are not exactly made in the same way. There are different HDF5 groups for the header and other parameters, *BUT* essential information, such as particles' positions and velocities, are stored in the same way.

Auxiliary programs and commands

Vitables

Usefull for messing with HDF5 files. I recomend always opening snapshots in "read-only".

```
sudo apt install vitables
```

ssh

It is used for connecting to an external server by using the terminal. Comes by default with most linux distros and mac systems.

Exemplo:

```
ssh <user>@<host>
```

If the username from the connecting computer is the same as the account being connected in the server, just

```
ssh <host>
```

is enough.

ssh keys

Sometimes, when downloading/uploding numerous files from a ssh connection, it gets cumbersome to always type the password. For this, you can generate a pair of keys that, when installed, will eliminate the need to always enter the password when connecting to the ssh server.

First the keys are created (a usual path is `~/.ssh`)

```
ssh-keygen -f <path>/<key_pair_name>
```

Then the keys are copied and installed onto the server

```
ssh-copy-id -i <path>/<key_pair_name> <user>@<host>
```

Now, depending on server and key settings, there shouldn't be the need to repeatedly type the password.

man

Shows a manual for a specific command. Can be used as

```
man <command_to_be_explained>
```

scp

Basically `cp`, but can be used to transfer files over a network. Can be used as

```
scp <origin> <destination>

scp <user>@<server>:<origin> <destination>

scp <origin> <user>@<server>:<destination>

scp <user>@<server>:<origin> <user>@<other_server>:<destination>
```

If the user is the same at the server and the computer issuing commands, `<user>@` can be ommited.

rsync

This command can recursively sincronize a set of files, usually being faster than `scp` to copy large quantities of them.

It can be used in a similar manner to `scp`

```
rsync -avP <origin> <destination>
```

where the options `-avP` would use the archive mode (`-a`) with verbose output (`-v`) and a progress bar (`-P`).

qsub, qstat, qdel

Basic commands for using the queue in hercules.

- `qsub` is used for subimitting the job. A `submit.job` file is usually given as parameter with specific properties and the command to be executed;
- `qstat` returns the current running jobs. The first column by default is the job id followed by the machine running the job.
- `qdel` can "delete" a job, sending a stop signal to the process. It is used as

```
        qdel <job id>
```

less

Useful command for poking around text based files. Can be used as

```
less <path_to_file>
```

Navigation is simply done with the page up/down, mouse wheel (on certain terminal emulators) and the arrow keys.

top and htop

Shows all tasks running in the system at the moment, per thread usage and other useful information. Simply used by issuing

```
top

htop
```

`htop` is a incremented version of `top` and is easier to use.

Author

Elvis de Mello

Github: https://github.com/elvismello

e-mail: `elvis.mello@outlook.com` / `elvis.mello@ufpr.br`