User Manual for Gevolab v0.1

Alireza Vafaei Sadr

April 24, 2017

Contents

1 Perface

Gevolab is an API for Gevolution[1] [1] powered by other packages to access and analyze data of Gevolution simulations. Gevolution is a particle-mesh N-body code which is based on general relativistic treatment of gravity. This package includes LATfield[2] [2] which provides a parallel framework for managing data, in particular Fields on a three dimensional Lattice. Some Gevolab functions also are boosted by using LATfield. Also this package is included two additional packages Rockstar Halo Finder[3] [3] and pyGadgetReader[4] [4] for more analysis over data in an integrated platform.

2 Setting up

2.1 Requirements

Since there may exist some version dependence bugs, it is highly recommended to use Anaconda distribution of the Python2.7. There are some needed packages for Gevolab.

- numpy[5]
- h5py[6]
- gcc-python-plugin[7]

In the case of using anaconda you can install them simple using:

```
[USER@system gevolab]$ conda install numpy
[USER@system gevolab]$ conda install h5py
[USER@system gevolab]$ conda install -c anaconda gcc
```

2.2 Installation

This version of Gevolab is tested by Python2.7 on Ubuntu distribution of Linux. Is is highly recommended to use Anaconda[8] distribution of the Python for less version dependent inconsistencies.

Also non-python packages may need libraries separately. So you can modify **setup.py**. There are two variables **includes** and **libs** for needed includes and libraries.

Gevolab will be installed using simple bellow command:

```
[USER@system gevolab]$ python setup.py install
```

where **python** is your own python interpreter.

[1]https://github.com/gevolution-code/gevolution-1.1
[2]http://www.latfield.org/
[3]https://bitbucket.org/gfcstanford/rockstar
[4]https://bitbucket.org/rthompson/pygadgetreader
[5]http://www.numpy.org/
[6]http://www.h5py.org/
[7]https://github.com/davidmalcolm/gcc-python-plugin/blob/master/docs/index.rst
[8]https://www.continuum.io/downloads

3 Package structure and functions

Source directory of package includes gevolab directory which contains all of classes and functions. Other needed (non-python) packages and data are static directory. You can modify that packages by replacing your own packages (maybe your specific modified version!).

3.1 *gev_analyze* class

There is a defined class in Gevolab that you can make an instance using it according to a Gevolution **setting.ini** file. A **setting.ini** file should contain all of necessary information about a Gevolution simulation. For example if you have a setting file in directory test, you can make an instance like:

```
from gevolab import gev

add = '/home/user/test/settings_sample.ini'
sample = gev.gev_analyze(add)
```

.

There are several methods to access and analyze chosen setting file outputs. Also you can run Gevolution within python according to the setting file.

Now assuming above code, we can use bellow methods:

1. *setting_getter()*: You can extract all of selected setting file information by using this method on the defined instance:

   ```
   print sample.setting_getter()
   ```

   .

2. *load(type,name,nsnap)*:

   This method can load either particle or field snapshots in *.h5* format also power spectrum files.
   Inputs:
   type is a string with two possible optiones. For snapshots you should use *'snap'* and for power spectrum, *'pk'*.
   name is the filed you wanted to load. It should be in string format and select from setting file defined options.
   nsnap is the number of requested snapshot.

   for example for loading third particles snapshot and T_{00} power spectrum for second snapshot you can use

   ```
   parts = sample.load('snap','pcls',3)
   power_t_00 = sample.load('pk','T00',2)
   ```

   respectively.

3. *local_run(n,m))*:

   This method is for running Gevolution using the specified options in selected setting file on your local computer. This method can be executed stand alone and does not need installed Gevolution. But in the case of running your modified version of Gevolution, you can put setting file in your version directory. Method can detect your version and run it automatically.
   Inputs:
   m and n are processor numbers of LATfield.

   For example for using 8 cores of a local computer

   ```
   sample.local_run(4,2)
   ```

   can be used.

4. *job_submit(n_core,time)*:

   This method is for running Gevolution using the specified options in selected setting file by submitting a job on cluster. It can generate run script and submit it.
   This method also can be executed stand alone and does not need installed Gevolution. But in the case of running your modified version of Gevolution, you can put setting file in your version directory. Method can detect your version and run it automatically.

   Note that this method is written according to dpt group of Baobab cluster. I can make it more flexible for other options according to what you want to change.

   Inputs:
   n_core
   Number of cores. Default value is 16. *time*
   Wall time limit of the script. Note that wall time should inputted as string. Default time is $'5:00:00'$.

   For example for running on 64 cores by $10:00:00$ wall time.

   ```
   sample.job_submit(n_core=64,time='10:00:00')
   ```

5. *rockstar(nsnap)*:

   This method runs rockstar halo finder on the selected number of snapshot. Note that, outputs will be saved as two files *halos_0.0.ascii* and *halos_0.0.bin*. So if you want to save them, you have to rename them after running to avoid replacing them next time. Also it is obvious that if you want to do rockstar analysis on your selected setting outputs, you have to add 'pslc' for snapshot outputs.

   Inputs:
   nsnap is the number of snapshot you want to run rockstar on.

   For example for first snapshot, you can simply write

```
sample.rockstar(1)
```

.

6. *rockstar_bin(key)*:

   This method can load features of found halos in rockstar binary output. So this method can be used only after running *rockstar* method. For more information you can take a look at Rockstar Halo Finder[9] [3] or pygadgetreader website[10] for more comprehensive examples.

   Inputs:
   key is the key of requested information.

   For example information of particle IDs of halos, you can write

   ```
   sample.rockstar_bin('particles')
   ```

   .

7. *duplicate(filename, **opts)*:

   This method can be used for generating a modified version of selected setting file within Gevolab. For example you want to run a first setting configuration and rerun it with several changes.

   Inputs:
   filename is the name of new setting file you wanted to save.
   ***opts* is(are) a python dictionary included the parameters you wanted to modify.

   For example if you want to change initial redshift and Ngrid, you can write

   ```
   new_setting_add = '/home/user/test/settings_sample2.ini'
   sample.duplicate(new_setting_add, **{"initial redshift":50\
   ,"Ngrid":128})
   ```

   .

8. *outputs_push(type,name,nsnap)*:

   This method can transfer output files from cluster to a defined address. If you want to use this method, you first have to set host path.
   Inputs:
   type is the type of requested output. It can be either 'pk' for power spectrum or 'snap' for snapshot.
   name is the filed you wanted to transfer. It should be in string format and select from setting file defined options

[9] https://bitbucket.org/gfcstanford/rockstar
[10] https://bitbucket.org/rthompson/pygadgetreader

for example if you want to transfer second snapshot of phi filed on your assumed local computer, you can write *nsnap* is the number of snapshot.

```
sample.hostpath = 'user@192.168.1.1:~/outputs/'
sample.outputs_push('snap','phi',2)
```

.

Every time you use this command, it will request for password, so you can use ssh key generators for more ease.

9. *push(filename)*:

   This method can transfer any desired file from cluster to a defined address. If you want to use this method, you first have to set host path. Also this method is almost independent from setting file and it is just defined to have more flexibility over files.

   Inputs:
   filename is the full path file name of desired file.

   ```
   sample.hostpath = 'user@192.168.1.1:~/outputs/'
   file_add = '/home/user/gevolution/setting.ini'
   sample.outputs_push(file_add)
   ```

10. *snap_movie()*:

    This method can make a iterative animation from outputted snapshots. You have to add 'pcls' to setting file to use this method.
    Using this method is simply

    ```
    sample.snap_movie()
    ```

    .

3.2 Functions

1. *extractor(file_name,id_add,outname,function)*:

   Inputs:
   file_name is the snapshot file that we want to use mask over.
   id_add is the address of particle IDs list file in ASCII format which you wanted to filter.
   outname is the full path of output snapshot file name.
   function is the transformation function for all 3 dimension positions. If you do not input this argument, default value will be set. Default is None and will use no transformation on particle positions.

```python
from gevolab import gev
import numpy as np

file_name = '/home/user/gevolution/output/lcdm_snap000_cdm.h5'
id_add = '/home/user/gevolution/ids'
outname = '/home/user/IC/ic.h5'

def function(x):
        x_mean = np.mean(x)
        xmin = np.min(x)
        xmax = np.max(x)
        if xmax-xmin>0.8:
                x = x+0.5
        x = np.mod(x, 1.)

        xmin = np.min(x)
        xmax = np.max(x)
        x = 1.2*(x-xmin)
        x_mean = np.mean(x)

        return x-x_mean+0.5

gev.extractor(file_name,id_add,outname,function)
```

2. *plot_snap_save(snap)*: This function can draw and save snapshot in *.jpg* format. If you want to run this function on servers, you have to use 'agg' mode of matplotlib.

   snap is the full path address of snapshot file.

   Here is an example of this function.

```python
import matplotlib as mpl
mpl.use('agg')

snap = '/home/user/gevolution/output/lcdm_snap000_cdm.h5'
gev.plot_snap_save(snap)
```

3. *snap_show(snap)*:

   This function can show a snapshot in an interactive frame. You can not run this function on servers.

   snap is the full path address of snapshot file.

   Here is an example of this function.

```python
import matplotlib as mpl
```

```
snap = '/home/user/gevolution/output/lcdm_snap000_cdm.h5'
gev.snap_show(snap)
```

References

[1] J. Adamek, D. Daverio, R. Durrer and M. Kunz, JCAP **1607**, no. 07, 053 (2016) doi:10.1088/1475-7516/2016/07/053 [arXiv:1604.06065 [astro-ph.CO]].

[2] D. Daverio, M. Hindmarsh and N. Bevis, arXiv:1508.05610 [physics.comp-ph].

[3] P. S. Behroozi, R. H. Wechsler and H. Y. Wu, Astrophys. J. **762**, 109 (2013) doi:10.1088/0004-637X/762/2/109 [arXiv:1110.4372 [astro-ph.CO]].

[4] R. Thompson, pyGadgetReader: GADGET snapshot reader for python, Astrophysics Source Code Library (2014).