

REPORT TFG-EXP
COSMOLOGICAL SIMULATIONS

Tomás Riera Gómez
Universidad Autónoma de Madrid

APPENDIX A: GADGET4 INSTALLATION

It is important to note that all the information about Gadget4 can be found in their official webpage : <https://wwwmpa.mpa-garching.mpg.de/gadget4/> . The official user manual can also be found following this link, in the “documentation” section.

WHAT YOU’LL NEED

Operating system

The first and most important requirement is that you are operating on a Linux environment. It might be possible to install the code in a different environment, but this guide is oriented towards the installation of the code in a Linux operating system, otherwise the following instructions won’t work.

Firstly, check everything is up to date by using the `sudo apt update` command. This will download and update the package information from all of the configured sources. Also, the fortran compiler and the library z are needed for the (mentioned below) libraries to work. This should be enough for us to be able to install the code.

Knowing the number of processors and cpu cores in your computer is also required, since for the simulations you need to specify the number of these that the code will use. To do this, type into the terminal `less /proc/cpuinfo`. Here you will find information about each of the processors, including the number of cores for each one.

Libraries

You will need to install (or have installed) some libraries as well in order to run the code:

- mpi: The ‘Message Passing Interface’ (version 3.0 or higher). Many vendor supplied versions exist, in addition to excellent open source implementations. We will be using MPICH (<https://www.mpich.org/>).
- gsl: The GNU scientific library. This open-source package can be obtained at <http://www.gnu.org/software/gsl>
- fftw3: The ‘Fastest Fourier Transform in the West’. This open-source package can be obtained at <http://www.fftw.org>
- hdf5: The ‘Hierarchical Data Format’ (available at <http://hdf.ncsa.uiuc.edu/HDF5>)

all these need to be installed in a certain order as follows:

- 1 mpi compiler
- 2 fftw and hdf5
- 3 gsl

The most useful way to do this is use a shell script that will unpack the compressed zip files and install them in a directory of the user’s choice. It is recommended to install all libraries in a folder together, so that everything is easier to locate.

Gadget4 source code

From the Gadget4 website (<https://wwwmpa.mpa-garching.mpg.de/gadget4/#gadget-4>) the user can download the gadget4 files in the “Obtaining the code” section, which takes you to this repository : <https://gitlab.mpcdf.mpg.de/vrs/gadget4> . Download all these files and keep them in a directory together, as this directory will be needed later for the compilation and usage of the code.

Having done this, the user will need to edit some files:

- In Makefile, specify the absolute path to python in their computer
- In Makefile.systype (the files include a template) add the your system type
- In Makefile.comp.gcc the user should set the path to the mpicxx compiler installed with MPICH
- In Makefile.gen.libs set the external library paths (mpi, hdf5, fftw, gsl...)

Now the user just needs to set some compile-time options before compiling the code.

SETTING UP THE FILES : CONFIGURATION FILE AND PARAMETER FILE

We now go to the directory where the user has all the gadget4 files (which were previously downloaded from the repository <https://gitlab.mpcdf.mpg.de/vrs/gadget4>). Before making (compiling) the actual program, i.e. the executable that will be used to run the simulation, Gadget4 requires a configuration file with compile-time options that specify different technical aspects of the code. Once compiled, Gadget4 requires a parameter file that has the same function for the run-time options. It is often easier to just write both, make the code and then run the simulation. These files contain different commands (combinations of characters) that, when included in the corresponding file, activate a certain feature within the code. In the official manual (<https://wwwmpa.mpa-garching.mpg.de/gadget4/>) a list of all the compile-time options (configuration file) and run-time options (parameter file) is provided. Some of these, of course, are required for the code to run, as they specify the type and characteristics of the simulation that is being performed. The configuration file has to be called *Config.sh*, and the parameterfile has to be in .txt format in order for the code to work. It might work for other file formats, but this is not guaranteed.

CREATING THE EXECUTABLE : MAKE AND MAKE CLEAN

Once all the compile-time options in Config.sh have been specified, the user can proceed to create the code by typing in the terminal the command **make** and pressing enter. This should create an executable file called *Gadget4*, which is the actual code that will run the simulation, created according to the flags specified in *Config.sh*. If the user wanted to change any of the compile-time options after compiling the code, it is not enough to just type them into the configuration file. This needs to be done of course, but also the code needs to be “un-compiled” and compiled again. Firstly, the user needs to type **make clean** into the terminal. This will undo everything the command **make** did, erasing the corresponding files and the executable. Once the configuration file is properly set up, the user can proceed to type into the terminal **make** again, compiling the code with the new compile-time options and creating a new *Gadget4* executable.

RUNNING A SIMULATION

Once the code has been compiled correctly and all the parameters have been set, the user needs to know the absolute path to the *mpiexec* file that was installed with mpich (within *bin/*) The simulation is started by typing into the terminal the following command (while in the Gadget4 directory, of course) :

```
[path to mpiexec] -n [number of mpi ranks to use] ./Gadget4 [name of parameterfile]
```

And pressing enter. The optimal number of mpi ranks to use so as not to saturate the computer is half the total number of cpu cores. For example, in a computer where mpiexec is in a directory with absolute path */home/tomi/Desktop/TFG/Software/Gadget4/libraries/build/bin/mpiexec*, 8 cpu cores in total (e.g. if we have 4 processors with 2 cpu cores per processor) and a parameterfile named *param.txt*, to start the simulation with the optimal number of mpi ranks the user would type into the terminal:

```
/home/tomi/Desktop/TFG/Software/Gadget4/libraries/build/bin/mpiexec -n 4  
./Gadget4 param.txt
```

And then press enter. The simulation would start to run, and would do so (and terminate) according to the configuration and parameters specified in *Config.sh* and *param.txt*. The output format of the simulations is also specified within these files, and will be stored in a (new) directory within the current directory, named *output/*. There is no need to create it, as it will be automatically generated to store the results of the simulation.