Tutorial: CLASS and MontePython Installation

Chakkrit Kaeonikhom and Nandan Roy

November 2023

1 Overview

[1]CLASS (Cosmic Linear Anisotropy Solving System) is a numerical tool for solving the evolution of linear cosmological perturbations and for computing the cosmological observables for a given input model. CLASS is a flexible and user-friendly code that can be easily generalized to non-minimal cosmological models. CLASS was written by Julien Lesgourgues & Thomas Tram, and first released in 2011.

CLASS is written in `C` language for each module. It comes with `C++` and `Python` wrapper.

For more information about CLASS can be found on the website: http://class-code.net. The CLASS papers can be found below

- CLASS I: Overview, by J. Lesgourgues, arXiv:1104.2932 [astro-ph.IM]
- CLASS II: Approximation schemes, by D. Blas, J. Lesgourgues, T. Tram, arXiv:1104.2933 [astro-ph.CO], JCAP 1107 (2011) 034

2 CLASS part

2.1 Pre-requistes

Mac users

`gcc` can be installed via Homebrew[2], following the command

```
$ brew install gcc
```

Next, we must install `Cython` which can be installed via

[1]Adapted from Deanna C. Hooper

[2]If no Homebrew installed in the system, one can find installation instruction from https://docs.brew.sh/Installation

```
$ pip install Cython
```

Linux users

You need to have `gcc` compiler `Python`, and `Cython`.

To install `gcc` use the following commands,

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install gcc
```

Almost every Linux distribution comes with a version of Python included in the default system packages. Check your Python version by using the following command;

```
$ python3 --version
```

or

```
$ python --version
```

In case you want a better handling of the Python packages download and install Miniconda or Anaconda from the following links;

Miniconda
https://docs.conda.io/projects/miniconda/en/latest/
Anaconda
https://www.anaconda.com/download
Next to install cython;

```
$ pip install Cython
```

or

```
$ conda install Cython
```

Downloading CLASS

We can get the latest version of CLASS by downloading via git using the command

```
$ git clone https://github.com/lesgourg/class_public.git
```

We can also download CLASS directly from Github homepage (see Figure 1).

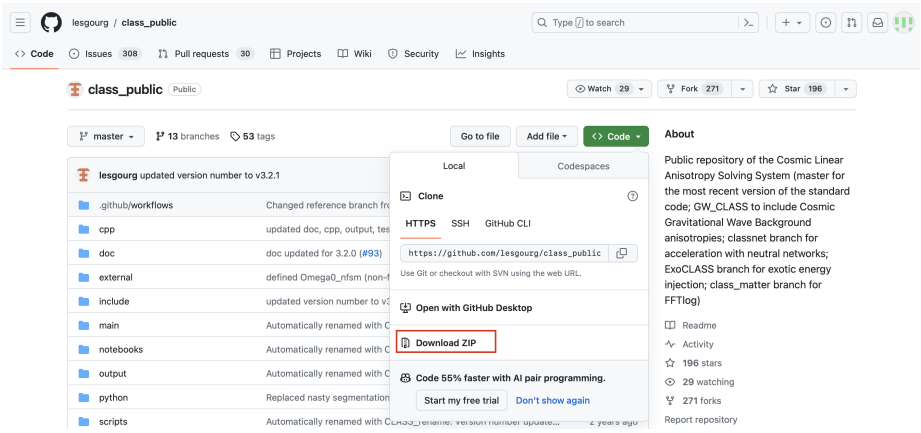

Figure 1: Screen short of CLASS homepage on Gishub

Installing CLASS

Go to CLASS directory via terminal. Compile `C` code and `Python` wrapper using the command

```
$ make -j
```

Mac users may have to install command line tool for xcode by going to the site: https://developer.apple.com/download/all/ (sign in is required), click download Command Line Tools for XCode (see Figure 2)

Execute the command

```
$ ./class explanatory.ini
```

to test if the `C` code installed successfully.

Setting Python wrapper

In the CLASS directory, go to the 'python' directory and execute the command:

```
$ python setup.py build
$ python setup.py install --user
```

Checking if the steps above worked fine by typing

```
$ python
>>> from classy import Class
```

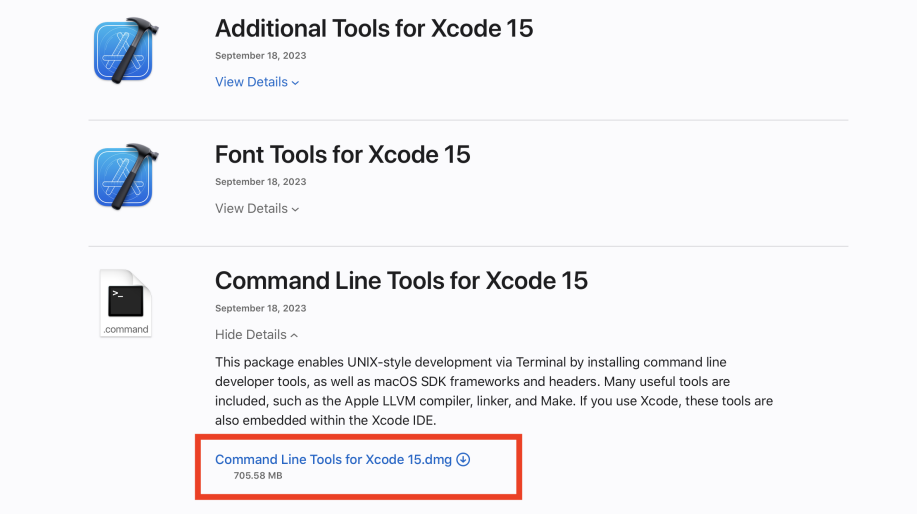

Figure 2:

3 Monte Python part

3.1 Getting Monte Python

MontePython can be downloaded by

```
$ git clone https://github.com/brinckmann/montepython_public.git
```

[3]You need the Python program version 2.7.x** or version 3.x**. Your Python must have 'numpy' (version >= 1.4.1) and 'Cython'. The last one is used to wrap CLASS in Python.

Optional: If you want to use the plotting capabilities of Monte Python fully, you also need the 'scipy', with interpolate, and 'matplotlib' modules.

After installation, go to the directory:

```
$ cd montepython_public
```

Make a copy of the file `default.conf.template` and rename it to `default.conf` (or any name you prefer)

```
$ cp default.conf.template default.conf
```

At minimum, the file `default.cof` needs one line:

[3]Adapted from Benjamin Audren

```
path['cosmo'] =  path/to/your/class_public
```

Make sure that your CLASS directory has the same name as in the path.

Now the code is installed. There are two main (optional) commands in `MontePython`: the first one is `run`, for running MCMC to generate chains, another one is `info`, for analysing chains. More information can be viewed by executing

```
$ python montepython/MontePython.py --help
$ python montepython/MontePython.py run --help
$ python montepython/MontePython.py info --help
```

To test a small running, executing:

```
path['cosmo'] =  path/to/your/class_public
```

To get running, type:

```
$ python montepython/MontePython.py run -o test -p example.param
```

If the directory 'test/' doesn't exist, it will be created, and a run with the number of steps described in 'example.param' will be started. To run a chain with more steps, one can type:

```
    $ python montepython/MontePython.py run -o test -p
    ↪  example.param -N 100
```

To analyse the chains, type

```
$ python montepython/MontePython.py info test/
```

For more information, see https://baudren.github.io/montepython.html. The documentation can be found in this link.

The MontePython papers can be found as

- Brinckmann, T. and Lesgourgues, J., "MontePython 3: Boosted MCMC sampler and other features", Physics of the Dark Universe, vol. 24, 2019. doi:10.1016/j.dark.2018.100260.

- Audren, B., Lesgourgues, J., Benabed, K., and Prunet, S., "Conservative constraints on early cosmology with MONTE PYTHON", Journal of Cosmology and Astroparticle Physics, vol. 2013, no. 2, 2013. doi:10.1088/1475-7516/2013/02/001.

4 Setting up Planck likelihoods

[4]The Planck 2018 data can be found on the 'Planck Legacy Archive http://pla.esac.esa.int/pla/#home. The Planck Likelihood Code (plc) is based on a library called 'clik'. It will be extracted, alongside several '.clik' folders that contain the likelihoods. The code uses an auto installer device, called 'waf'. Here is the detail of the full installation.

Next, create a directory which you want to store Planck data, and go into that directory

```
$ mkdir -p path/to/planck && cd $_
```

Download the code and baseline data (will need 300 Mb of space)

```
$ wget -O COM_Likelihood_Code-v3.0_R3.01.tar.gz
↪  "http://pla.esac.esa.int/pla/aio/product-action?COSMOLOGY.F
↪  ILE_ID=COM_Likelihood_Code-v3.0_R3.01.tar.gz"

$ wget -O COM_Likelihood_Data-baseline_R3.00.tar.gz
↪  "http://pla.esac.esa.int/pla/aio/product-action?COSMOLOGY.F
↪  ILE_ID=COM_Likelihood_Data-baseline_R3.00.tar.gz"
```

Uncompress the code and the likelihood, and do some clean-up

```
$ tar -xvzf COM_Likelihood_Code-v3.0_R3.01.tar.gz
$ tar -xvzf COM_Likelihood_Data-baseline_R3.00.tar.gz
$ rm COM_Likelihood_*tar.gz
```

Move into the code directory

```
$ cd code/plc_3.0/plc-3.01
```

Configure the code. Note that you are **strongly advised** to configure `clik` with the Intel mkl library, and not with `lapack`. There is a massive gain in execution time: without it, the code is dominated by the execution of the low-l polarisation data. Before the next step, ensure you do NOT have any old Planck likelihoods sourced!

```
$ ./waf configure --lapack_mkl=${MKLROOT} --install_all_deps
```

If everything goes well, you will be ready to install the code

```
$ ./waf install
```

Next, source the likelihood. If you are running on a shell, type the command

[4]Adapted from Deanna C. Hooper

```
$ source bin/clik_profile.sh
```

Running on a z-shell, you will first need to create a .zsh version of the above file. This can be done in many ways, for example

```
$ cp bin/clik_profile.sh bin/clik_profile.zsh
$ sed -i 's/addvar PATH /PATH=$PATH:/g' bin/clik_profile.zsh
$ sed -i 's/addvar PYTHONPATH /PYTHONPATH=$PYTHONPATH:/g'
↪  bin/clik_profile.zsh
$ sed -i 's/addvar LD_LIBRARY_PATH
↪  /LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/g' bin/clik_profile.zsh
$ source bin/clik_profile.zsh
```

You need to add

```
$ source
↪  /path/to/planck/code/plc_3.0/plc-3.01/bin/clik_profile.sh
```

to your `.shellrc` (or the `.zsh` to your `.zshrc` on a z-shell), and you should put it in your scripts for cluster computing.

After successfully installing Planck likelihoods, in MontePython configuration file, you will need to add

```
path['clik'] = '/path/to/planck/code/plc_3.0/plc-3.01'
```

There are nine Planck 2018 likelihoods defined in *Monte Python*: `'Planck_highl_TT'`, `'Planck_highl_TT_lite'`, `'Planck_highl_TTTEEE'`, `'Planck_highl_TTTEEE_lite'`, `'Planck_lensing'`, `'Planck_lowl_TT'`, `'Planck_lowl_EE'`, `'Planck_lowl_EEBB'`, `'Planck_lowl_BB'`, as well as five sets of parameter files, bestfit files, and covmats.