MontePython Exercises
MontePython + CLASS Kavli workshop
Cambridge, 2018

Thejs Brinckmann (courtesy of Miguel Zumalacarregui)

September 12, 2018

Running time scales: These exercises only involve fast runs that can be done on a laptop in few minutes, excepted exercise 3 which has more parameters and may require a few hours of running: thus we recommend to start exercise 3 in the first session (but after doing exercise 1, in order to get more familiar with the code), and to analyze the results of this run in the second exercise session. If you cannot keep your laptop running for a few hours after the first exercise session, try to launch the run of exercise 3 on a remote desktop or a cluster.

MontePython with parallel chains: if your laptop, desktop or cluster does not have `MPI` pre-installed, do not despair! In this case you should just launch multiple chains with separate executions of `MontePython` pointing to the same chains directory. There will still be communication between the chains through small files, with no perceptible loss in efficiency, even with periodic covariance matrix update and running jumping factor adaptation.
In both cases, you can speed up each chain by running it on a few cores with e.g. `OMP_NUM_THREADS=4` (the `CLASS` parallelization is efficient up to about 8 cores).

Exercise 1: The $\Omega_m - \Omega_\Lambda$ plane

This exercise relies on constraints from the cosmic expansion: is fast enough to be done on a laptop ($\sim$ 0.1s/model on mine). You can let it run for a few minutes while looking at another exercise, come back to it and make some cool plots!

We will obtain Baryon Acoustic Oscillation (BAO) constraints on the $\Omega_m - \Omega_\Lambda$ plane for a two parameter model. We will vary Ω_{cdm} and Ω_k, keeping Ω_b and H_0 fixed by setting the 1σ values to zero (recall Ω_Λ is a derived parameter). Use the following example:

```
data.experiments=['bao_boss_dr12','bao_smallz_2014']
data.parameters['Omega_cdm'] = [0.3, 0, None, 0.05, 1, 'cosmo']
data.parameters['Omega_k'] = [0.0, -0.5,0.5, 0.05, 1, 'cosmo']
data.parameters['Omega_Lambda'] = [1,None,None,0, 1, 'derived']
data.cosmo_arguments['Omega_b'] = 0.048
data.cosmo_arguments['h'] = 0.68
data.N=10
data.write_step=5
```

It is important to give meaningful names to the files and folders to keep track of your work. I suggest labeling this combination of parameters and experiments as `lc_bao.param` (this notation means `l` for Λ, `c` = CDM and `bao` = BAO from BOSS galaxies).
Note: the limits on Ω_k are such that `CLASS` does not complain.

The basic part of the exercise involves the following steps:

a) Write the above into a `.param` file (⚠ if you copy/paste from this pdf to a text file, the ′ sign will appear wrongly in the text file: you will have to replace it manually by a regular vertical single bracket). Do a short Monte Python run:

```
time python montepython/MontePython.py run -p input/lc_bao.param -o chains/lc_bao -N 10
```

The `time` prefix is just to know how long it will take (you can use that information to adjust the parameters to how much time you have).

b) Now you can do the serious run with four chains. If you are on a cluster or a machine with `mpirun` installed, you can type

```
mpirun -np 4 python montepython/MontePython.py run -o chains/lc_bao -N 10000
```

but if you are on your laptop, no problem, just launch the four chains one after each other, by typing four times

```
python montepython/MontePython.py run -o chains/lc_bao -N 10000 --silent &
```

In both cases, the chains will exchange information to speed up convergence (since the option `--update 50` is active by default). Note that because you are running from a folder with a `log.param` you do not need to provide the `.param` file again. The `--silent` option avoids an unreadable accumulation of plethoric output from your four chains on a single terminal.

c) Relax while your computer does the work. At any time you can check the chain size with `wc -l chains/lc_bao/*.txt`. If you are impatient and there are enough points, you can stop the run (`Ctrl+c` or four `kill` commands) and proceed. Because we have only two varying parameters, one or two thousands per chain will already give nice plots, three thousands is a luxury.

d) Analyze the chains. You can use MP in `info` mode:

```
python montepython/MontePython.py info chains/lc_bao --want-covmat
```

plus the optional options. Take at the nice plots in `chains/lc_bao/plots` and at all the other files generated in the analysis. Eventually, the covariance matrix file `lc_bao.covmat` and bestfit file `lc_bao.bestfit` could be passed in input of another similar run (with `-c` and `-b`) in order to speed up convergence.

e) You are encouraged to play with the different options. Try to plot the marginalized contours using $\Omega_m = \Omega_{cdm} + \Omega_b$ instead of Ω_{cdm}. This can be done with a customisation file `--extra plot_file/my_customization.plot`, and in particular by playing with the commands `info.redefine` and `info.to_change` documented in `plot_file/example.plot` .

f) If you do other runs with some parameters in common but with other datasets (some ideas are suggested below), you can plot them together. Just write the list of output folders after `info` . How do the different constraints compare?

⚠ Make sure that each model/data combination goes to a different `-o` directory! Otherwise you'll keep running the same thing over and over again.

The rest of the exercise is optional and slightly harder. Bear in mind that the amount of free parameters increases and the runs will require more time.

More realistic BAO analysis: By not varying Ω_b, H_0 we are fixing the comoving BAO scale r_s. Although r_s is well constrained by the CMB, there is some variability, which you may take into account by letting Ω_b vary within some range.

Do a run varying the baryon fraction. You can do this in two ways

a) More elegant: add a gaussian prior on $\omega_b \equiv \Omega_b h^2 = 0.02222 \pm 0.00023$, or equivalently, a gaussian likelihood. This is the goal of exercise 2.

b) Easier: add a top-hat prior to allow for 2σ deviations (with an appropriate input line of the type `data.parameters['omega_b'] = [...]` in the `.param` file, while the line fixing `Omega_b` should be commented out).

Supernovae: There are other background observations besides BAO, like type 1A Supernovae (SNe). Run the same model with the Union SNe compilation with `data.experiments=['sn']` (slightly obsolete data, but easy to run).
For a more challenging option you can use the JLA SNe sample `data.experiments=['JLA']`. You need to download the data, the `numexpr` package and add several nuisance parameters. Read the instructions in the likelihood folder and `jla.param`.

Exercise 2: Adding a new likelihood

Adding a new likelihood in Montepython is only as complex as the likelihood itself. You will get to see with this simple example.

Our goal is to add a gaussian prior on the physical baryon density using the Planck result

$$\omega_b = \Omega_b h^2 = 0.02222 \pm 0.00023\,, \tag{1}$$

(`https://arxiv.org/abs/1502.01589`, Table 1, col 6). The steps are:

a) Copy a simple likelihood folder from `montepython/likelihoods` (for instance `hst`) and rename it as `cmb_baryon`. Change the name of the `.data` file to be `cmb_baryon.data`.

b) In `cmb_baryon.data` change `hst`→`cmb_baryon` and `h`→`omega_b`. Update the central value and standard deviation according to eq. (1).

c) Update `__init__.py` by changing the name of the class, the data (as given `.data` file, it is read as `self.xxx`) and the theoretical value (`cosmos.omega_b()` as given by `classy`)

You can launch a short run with

```
data.experiments=['cmb_baryon']
data.parameters['omega_b'] = [2, 0, None, 0.02, 1e-2, 'cosmo']
data.N=10
data.write_step=5
```

and check that the chains are roughly gaussianly distributed around the mean (note that the parameter is rescaled by `1e-2`).

Exercise 3: Constraining σ_8

We now want to compute a posterior probability distribution on σ_8. This parameter can be treated by `CLASS` either as an input parameter (and then A_s is inferred by a shooting method and should not be passes simultaneously as an input parameter), or as a derived parameter (and then A_s should be an input parameter). The first approach is more interesting because it will guarantee that we sample σ_8 with a flat prior.

a) In order to constrain σ_8 we need datasets that constrain the amplitude of the matter power spectrum, for example `sdss_lrgDR7` and/or `kids450_qe_likelihood_public`. The SDSS galaxy clustering likelihood works out of the box, whereas the recent KIDS-450 weak lensing likelihood by Fabian Köhlinger requires downloading the data externally and changing the path to the data in the `.data` file. For instructions on using KIDS-450 see the readme in the likelihood folder for instructions.

b) We suggest to vary the following parameters (the most important one for the matter power spectrum): σ_8, n_s, h, $\omega_{\rm cdm}$, while ω_b can be kept fixed to 0.02222. Keep the likelihoods of exercise 1 (`bao_boss_dr12, bao_smallz_2014,` eventually supernovae) in order to get some additional constraints on $\omega_{\rm cdm}$ and h, and add the likelihood `hst` (direct measurement of H_0) to have one more independent constraint on h. Finally, to ensure convergence, it is wise to put at least some loose top-hat prior bounds on n_s, for instance $[0.8, 1.0]$.

This longer run is a good occasion to try running with the option `--superudate 20` , in order to speed up convergence.