MontePython Exercises
IFT School on Cosmology Tools

Miguel Zumalacarregui

March 16, 2017

The exercises have been ranked by level of difficulty (boring = 🧶, interesting = 🧶🧶, challenging = 🧶🧶🧶) and whether they require `hi_class` (`h_c`). Exercises marked 🕒 require short MCMC execution time and have parts suitable for execution on a laptop (do when varying 1-2 parameters, otherwise use Hydra).

MP in Hydra: To run MP in Hydra you will need to do the following:

1. Load the necesary modules in your script

   ```
   module load numpy
   module load scipy
   ```

2. For some reason (?) MontePython has an error when runninng as is, but it works in the LUSTRE filesystem. You have a folder asigned in that filesystem under

   ```
   /lustre/school/studXX/
   ```

   Make sure that the output directory (`-o`) is situated there, i.e. launch with

   ```
   python montepython/MontePython.py run -p your_model.param \
                              -o /lustre/school/studXX/your_model
   ```

3. For (optional) paralelization, add `prun` to your call (i.e. `prun python montepython/...`). You could in principle use `module load mpi4py` but that's not necessary (and makes Hydra complain).

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 (~ 0.1s/model on mine). You can let it run during the coffee break and make some cool plots when you come back!.

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','bao_boss_aniso']
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_b'] = [0.045, 0, None, 0.0, 1, 'cosmo']
data.parameters['h'] =      [0.68, 0, None,  0, 1, 'cosmo']
data.parameters['Omega_Lambda'] = [1,None,None,0, 1, 'derived']
data.cosmo_arguments['YHe'] = 0.24
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 so that CLASS does not coplain. Fixing the Helium fraction Y_{He} is good to run other cases, like SNe constraints varyinb Ω_b as well.

The basic part of the exercise (🧦) involves the following steps:

a) Write the above into a `.param` file and do a short Monte Python run:

```
time python montepython/MontePython.py -p 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

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

where you can adjust `N` to a larger number (`-N 10000` should be feasible, if you take a longer break you can increase accordingly). Because you are running from a folder with a `log.param` you don't need to provide the `.param` file again.

⚠ Add either `--update 300` (to improve the covariance matrix on the fly) or provide a covariance matrix with `-c name.covmat` (MP will interpret the parameters for you).

The call `mpirun -np 4` (use `prun` in Hydra, number of processes automatically determined) at the begining of the command runs 4 chains in the same console (adjust `-np` to your number of cores). If you don't have `openmpi` you can remove `mpirun` and just run on `-np` different consoles.

c) Relax while your computer does the work. If you come back early and there are enough points (go to output directory and type `wc -l *.txt` to count) you can cancell the work (press `Ctrl+c`).

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

```
python montepython/MontePython.py info chains/lc_bao
```

plus the optional options. Take at the nice plots in `chains/lc_bao/plots` and all the other files generated in the analysis.

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}.

f) Once you have several datasets (some ideas are suggested below) you can plot them together. Just feed MP `info` mode with several folders. 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.

Realistic BAO: By not varying Ω_b, H_0 we are fixing the comoving BAO scale r_s (i.e. the *coordintate* size of the standard ruler). 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 $\omega_b \equiv \Omega_b h^2$. This is the goal of exercise 2.

b) Easier: dd a hard prior to allow for 2σ devaitions (change `Omega_b`→ `omega_b` in the `.param` file, as well as the central value and limits).

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=['sne']`.
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: Modified gravity constraints (🎼🎼, h_c)

You implemented a model in `hi_class`, now you can test whether it makes sense or not!

Run an MCMC for the model you implemented in CLASS/`hi_class` sheet exercise 4. Run varying one of the α's parameters and the exponent n (see the end of exercise 4 for the syntax of `gravity_parameters_smg` in the `.param` file). For simplicity you can use 'synthetic' likelihoods like `fake_planck_bluebook`,`euclid_pk`, `euclid_lensing` and/or `fake_desi`. What happens to the constraints when n becomes very small? What happens when it's large?

Exercise 4: Running of the Planck Mass (🎼🎼🎼, h_c, 🕐)

This exercise is meant to use `hi_class` interfaced with Monte Python while using only background constraints, similarly to the previous exercise. Because the models currently implemented do not affect the background expansion, you will need to make some minor changes.

The first part of the exercise is to introduce a model with non-trivial background dynamics in `hi_class`. The Friedmann equations in `hi_class` (see `background_gravity_functions`) it read

$$H^2 = \rho\,, \qquad H' = -\frac{3}{2}a\,(\rho + p)\,, \qquad \texttt{(hi\_class)}\,, \tag{2}$$

where ρ is the total energy density, including the contribution from modified gravity. Recall that $H = \dot{a}/a$ (derivative wrt cosmic time, while $F' = \dot{F}/a$ denotes derivative wrt conformal time) and in CLASS the prefactor $8\pi G/3$ is absorbed in the units of ρ.

A more interesting model amounts to letting the effective Planck mass M_*^2 determine the cosmological strength of gravity by entering the Friedmann equation explicitly:

$$H^2 = \frac{\rho}{M_*^2}\,, \qquad H' = -\frac{3}{2}a\frac{\rho+p}{M_*^2}\,, \qquad \texttt{(modified)} \tag{3}$$

so that $M_*^2 > 1$ weakens the gravitational strength (i.e. $G_{\rm eff} = M_*^{-2}$). This model is described in Z. Huang's `https://arxiv.org/abs/1511.02808`, along with observational constraints, some of which we can reproduce.

The models (2, 3) are related by a redefinition of the dark energy density entering in ρ. By comparing both sets of equations we see that they are related by the following redefinition

$$\rho_{\rm smg} \longrightarrow \rho_{\rm smg} + \frac{1-M_*^2}{M_*^2}\rho\,, \tag{4}$$

where ρ is the total energy density (including ρ_{smg}), and similarly for the pressure. The existence of such a simple redefinition is why we have decided to keep the standard Friedmann equations for parameterizations (not to mention things like compatibility with the perturbation equations, or the cosmic sum rule). If you believe (and it is a reasonable belief) that (3) is the right description for modified gravity, or you just want to explore its consequences, you can just define a new `expansion_model_smg`.

a) Start by including eq. (4) in an expansion model. To keep things general we can use the $(w_0 - w_a)$ parameterization as the base (`wowa` in the code). In `background_gravity_functions` (of `background.c`) search for `wowa` and add

```
pvecback[pba->index_bg_rho_smg] += (1.-M_pl)/M_pl*(pvecback[pba->index_bg_rho_smg] +
    pvecback[pba->index_bg_rho_tot_wo_smg]);
pvecback[pba->index_bg_p_smg] += (1.-M_pl)/M_pl*(pvecback[pba->index_bg_p_smg] +
    pvecback[pba->index_bg_p_tot_wo_smg]);
```

Add a message to `background_gravity_parameters` in `case wowa`, so you know that you are modifying the Friedmann equation.

Better implementation: Instead of just modifying an existing parameterization, add an additional `expansion_model_smg` following the guidelines given in the `hi_class` exercise 4 (name suggestion `wowa_Mrun`). This will take a bit longer, but allows (2) and (3) to coexist.

b) Compile and make sure that the code works. Run it with the following `.ini` file

```
Omega_fld = 0
Omega_Lambda = 0
Omega_smg = -1
expansion_model = wowa #or your new expansion model name
expansion_smg = 0.7, -1, 0
gravity_model = propto_omega
parameters_smg = 1., 0., 1.0., 0., 1.
root = output/Mpl_run_test_1.0_
write background = yes please
```

Plot the angular diameter distance. You should find a $\approx -2.5\%$ decrease for $c_M = 1$ relative to $c_M = 0$. Although (3) naïvely suggests that a higher M_*^2 reduces the Hubble rate (which would *increase* the distances), the parameters are adjusted so the value of H_0 is equal in all cases and hence $H(z)$ increases with α_M at intermediate redshifts.

⚠Note that we're interested mainly in the background. However, with the modifications you have just made all linear perturbations (C_l, $P(k)$, etc...) are automatically consistent with the model.

c) Now test the model against the data! Vary the value of c_M for BAO and/or SNe. I suggest that you start by varying only α_M, with all the other parameters fixed. For the `.param` file try:

```
data.cosmo_arguments['expansion_model'] = 'wowa'
```

```
data.parameters['expansion_smg__1'] = [0.7, 0,None, 0.0, 1,  'cosmo']
data.parameters['expansion_smg__2'] = [ -1, 0,None, 0.0, 1,  'cosmo']
data.parameters['expansion_smg__3'] = [0.0, 0,None, 0.0, 1,  'cosmo']
data.cosmo_arguments['gravity_model'] = 'propto_omega'
data.parameters['parameters_smg__1'] = [1., 0, None, 0, 1, 'cosmo']
data.parameters['parameters_smg__2'] = [0., None,None, 0, 1, 'cosmo']
data.parameters['parameters_smg__3'] = [0., -2 ,2 , 1e-1, 1, 'cosmo']
data.parameters['parameters_smg__4'] = [0., 0, None, 0, 1, 'cosmo']
data.parameters['parameters_smg__5'] = [1., None,None, 0, 1, 'cosmo']
```

d) Plot the results and compare with `https://arxiv.org/abs/1511.02808`.

e) To make things more interesting repeat the run with different datasets and parameters.